

# Representing expressions in the semantic memory

Peter Schodl  
Arnold Neumaier

*Fakultät für Mathematik, Universität Wien  
Nordbergstr. 15, A-1090 Wien, Austria  
email: Peter.Schodl@univie.ac.at  
email: Arnold.Neumaier@univie.ac.at  
WWW: <http://www.mat.univie.ac.at/~neum/FMathL.html>*

## Contents

1	Introduction	1
2	The representation	2
3	Types of expressions	2
4	The typesheet for expressions	5
5	Not yet implemented	10
6	Examples of expressions	10

**Acknowledgements.** Earlier versions of this paper were discussed in meetings of the FMathL project. In particular, Kevin Kofler and Hermann Schichl contributed significantly through their remarks.

Support by the Austrian Science Fund (FWF) under contract number P20631 is gratefully acknowledged.

## 1 Introduction

This paper assumes [1]. We aim to represent in a transparent fashion all possible mathematical expression in the semantic memory.

An **operation** is anything that can be applied to mathematical expressions  $E_1, E_2, \dots$  such that the result  $E$  is an expression again. We call  $E_1, E_2, \dots$  the **subexpressions** of  $E$ . In particular, all standard functions, binary operations, and relations are operations, and so are quantification, merging expressions to form a set, a vector, etc. The operations are those categories that match the category **Expression** in the sense of [1].

We store the information in a fashion inspired by automatic differentiation, meaning we proceed from the most elementary subexpressions (its variables and constants) to the more complicated subexpressions by applying operations until the expression is fully covered.

## 2 The representation

Let the record `#handle` contain the expression  $E$ . Then we say that `#handle` is the **handle** of  $E$ . From the handle of some expression, the expression  $E$  itself and the free variables of  $E$  have to be accessible easily from `#handle`. The nodes representing the free variables of  $E$  are stored in `#handle.free` in the following fashion: For every node `#var` representing a free variable of  $E$ , we have `#handle.free.#var=#var`. If some expression does not have any free variables, then `#handle.VAR` is nonempty but does not have children.

The expression itself is constructed from its subexpressions in a recursive way, with constants and variables being expressions without subexpressions. The operation that is applied to the subexpressions of  $E$  is represented in the object `#handle.type`, the same object that is used for the typing of `#handle` (see [1]). How the subexpressions of  $E$  are represented in relation to `#handle` depends on the kind of operation, see below.

When an operation is applied to subexpressions, the free variables of the combined expression form the union of the free variables of the subexpressions, minus the variables that are bound by the operation. Every variable `#var` that is bound by application of the operation represented in `#handle.type` is stored as `#handle.binds.#var=#var`.

## 3 Types of expressions

We now illustrate the different types of expressions: since we build up all expressions from variables, constants and the application of operations to subexpressions, we have to describe the representation of these.

The handle of the expression is always denoted by `#handle` or `#h`.

### 3.1 Constants

There are currently three types of constants: strings, integers and floats, and all of them are represented in a similar fashion. The actual constant is always stored as the value of the handle of the constant. The record `#h` representing a constant has `#h.type=String` if `#h` is a string, `#h.type=Integer` if `#h` is an integer, and `#h.type=Float` if `#h` is a float.

For example, the string “Hello world” is represented as

$$\#h \xrightarrow{\text{type}} \text{String}$$

where `#h` is some anonymous node with `VALUE(#h) = Hello world`.

The type declaration of the constant types are:

```
String, Integer, Float:  
nothingElse>
```

The MATLAB creation code to create a string “Hello world” in record `string1` is

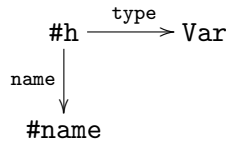
```
string1 = createstring('Hello world');
```

The corresponding commands for integers and floats are `createint` and `createdouble`.

### 3.2 Variables

The record `#h` representing a variable has `#h.type=Var`.

A name can, but need not be assigned to the variable. A variable with name `x1` is represented as



where the object `#name` is a string containing `x1`.

The type declaration of variables are:

```
Var:
optinal> name=String
nothingElse>
```

The MATLAB creation code to create a variable with name `x1` in record `variable1` is

```
variable1 = createvar('x1');
```

### 3.3 Operations with fixed arguments

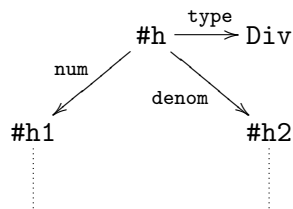
These are the operations that only allow a certain number of subexpressions to be applied to, and these subexpressions have a known role in the resulting expression  $E$ .

For example, the operation “squareroot” has one argument, the *radicand*, a fraction has two arguments, the *numerator* and the *denominator*, etc.

This reflects in the way these expressions are represented. For an expression  $E$  represented as record `#h` the subexpressions will be represented in `#h.#field` where the name of `#field` will usually unambiguously clarify the role of the subexpression in `#h.#field` for the expression in `#h`.

For example, consider an expression  $E$  with  $E_1$  being it numerator and  $E_2$  the denominator, hence  $E = \frac{E_1}{E_2}$ .

If  $E_1$  is represented in `#h1` and  $E_2$  is represented in `#h2` then the representation of the expression  $E$  in record `#h` (omitting the free variables) is:



The type declaration of a division is:

```
Div:
all10f> num=Expression, denom=Expression
```

The MATLAB creation code to create the division of the expression  $E_1$  in record `expr1` and expression  $E_2$  in record `expr2` to be stored in record `division1` is

```
division1 = create('Div',expr1,expr2);
```

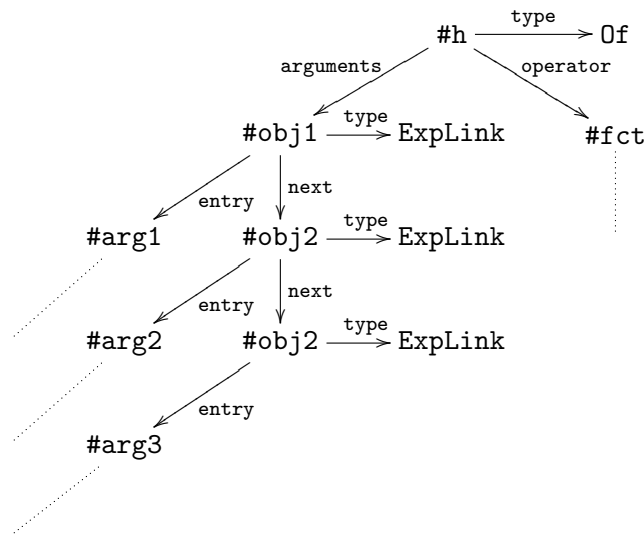
where the order of the arguments is significant: the first argument always contains the type, the

### 3.4 General n-ary Operations

Another kind of operations are those that admit an arbitrary number of subexpressions to be applied to, but all of these are treated equally. But there may still be a known number of subexpressions aside of these that have a fixed role.

For example, a case distinction between  $n$  cases, and as an extra argument the case “otherwise”, or the application of a function  $f$  to  $n$  arguments. In these cases, the  $n$  arguments are always represented as a linked list.

For example, consider the expression  $f(x_1, x_2, x_3)$  where  $f$  is represented in `#fct` and  $x_i$  is represented in `#argi`. Then the representation of the expression  $f(x_1, x_2, x_3)$  in record `#h` (again omitting the free variables) is:



The type declaration of this application is:

```
Of:
allOf> operator=Expression, arguments=ExpLink
```

```
ExpLink:
allOf> entry=Expression
optional> next=ExpLink
```

The MATLAB creation code to create  $f(x_1, x_2, x_3)$  with  $f$  in record `func1` and  $x_1, x_2, x_3$  in records `arg1`, `arg2`, `arg3` respectively in record `functionapplic1` is

```
functionapplic1 = create('Of',func1,{arg1,arg2,arg3});
```

### 3.5 Example

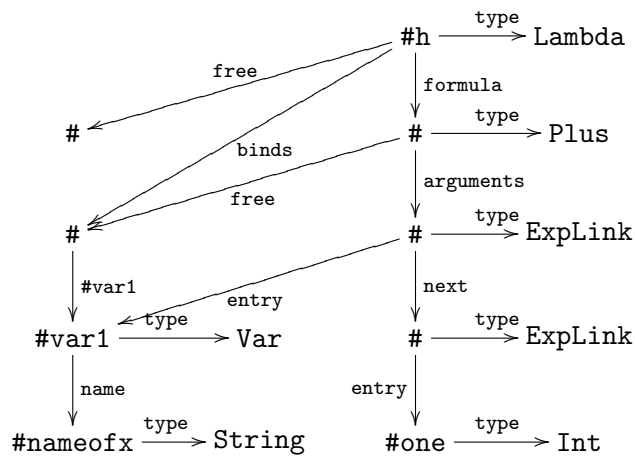
We give a complete record, with the bound variables and the free variables.

Consider the expression:

$$\lambda x.x + 1$$

The subexpressions are the variable  $x$  and the constant 1, the result of the operator Plus to these, resulting in  $x + 1$  (having free variable  $x$ ) and lastly the application of the operator Lambda binding variable  $x$ , hence resulting in  $\lambda x.x + 1$  which has no free variable.

Anonymous nodes that are not referred to are simply denoted by #. Assume that  $\text{VALUE}(\#one) = 1$  and  $\text{VALUE}(\#nameofx) = x$ .



## 4 The typesheet for expressions

The following is the typesheet that defines all the types that can be considered as operators to form expressions:

```
ExpressionTypes(TypeSystem)::
```

```
! Expression types
! -----
!
! Peter Schodl
!
! November 10, 2010
!
! This type system defines the types needed to process expression.
!
! Spacing is such that the file prints well with
!   lpr ExpressionTypes.cnt
!
```

```
EXPRESSION:
```

```
union> AND, ABS, BINRELATION, BRACKET, CASES, CHAIN, CARD
union> COMPOSITION, CONGRUENT, DISPLAY, DIAG, DIV, DUMMY, EVAL
union> EXISTS, EQUATION, ELLIPSIS, FORALL, FUNCTION, FLOOR, INVISMULT
union> INTEGER, INDEX, INTERSECTION, LAMBDA, LOG, MAX, MIN, MID
```

```
union> MINUS, MATRIX, NORM, NOTIN, NEGATIVE, OF, OTHERINTERVAL
union> PARTIAL, PARENTHESIS, PRIME, POWER, PROB, PLUS, ROOT, SQRT
union> SET2EXP, SUM, SUBSET, SUPERSET, SETMINUS, TIMES, UNARYRELATION
union> UNION, VARLIST, VECTOR, SETBUCKET, INTERVAL, INT, FORMULA
union> ORDEREDPAIR, SET, EMPTY, STRING, RELATION, VAR, LIST, UNEQUAL
```

AND:

```
someOfType> Integer = PROPOSITION
```

ABS:

```
allOf> entry = EXPRESSION
```

BINRELATION:

```
allOf> first = EXPRESSION, second = EXPRESSION
```

BRACKET:

```
allOf> entry = NODE
```

CASES:

```
someOfType> Integer = CASE
```

```
optional> otherwise = NODE
```

CASE:

```
allOf> formula = EXPRESSION, condition = GEN_STATEMENT
```

CHAIN:

```
allOf> firstrel = EXPRESSION
```

```
someOfType> Integer = NODE
```

CARD:

```
allOf> entry = EXPRESSION
```

UNEQUAL:

COMPOSITION:

```
someOfType> Integer = EXPRESSION
```

CONGRUENT:

```
allOf> lhs = EXPRESSION, rhs = EXPRESSION, mod = EXPRESSION
```

DISPLAY:

```
allOf> entry = EXPRESSION
```

DIAG:

```
someOfType> Integer = EXPRESSION
```

DIV:

```
allOf> nom = EXPRESSION, den = EXPRESSION
```

DUMMY:

```
allOf> entry = NODE
```

EVAL:

```

allOf> formula = EXPRESSION, BINDS = VAR
someOf> index = RELATION, from = EXPRESSION, to = EXPRESSION

EXISTS:
allOf> formula = EXPRESSION, scopedvar = EXPRESSION, BINDS = VAR

EQUATION:
allOf> formula = EXPRESSION
optional> where = GEN_STATEMENT

ELLIPSIS:
optional> ellipsed = NODE

FORALL:
allOf> formula = EXPRESSION, scopedvar = EXPRESSION, BINDS = VAR

FUNCTION:
allOf> from = EXPRESSION, to = EXPRESSION

FLOOR:
allOf> entry = EXPRESSION

INVISMULT:
someOfType> Integer = EXPRESSION

INDEX:
allOf> formula = EXPRESSION
someOf> sub = EXPRESSION, sup = EXPRESSION, lsup = EXPRESSION
someOf> lsub = EXPRESSION

INTERSECTION:
someOfType> Integer = EXPRESSION

LAMBDA:
allOf> formula = EXPRESSION, variable = EXPRESSION, BINDS = VAR

LOG:
allOf> entry = EXPRESSION

MAX:
allOf> formula = EXPRESSION, BINDS = VAR
optional> index = EXPRESSION

MIN:
allOf> formula = EXPRESSION, BINDS = VAR
optional> index = EXPRESSION

MID:
allOf> lhs = EXPRESSION, rhs = EXPRESSION

MINUS:
someOfType> Integer = EXPRESSION

```

MATRIX:  
someOfType> Integer = EXPRESSION

ROW:  
someOfType> Integer = EXPRESSION

NORM:  
allOf> formula = EXPRESSION  
optional> index = EXPRESSION

NOTIN:  
allOf> lhs = EXPRESSION, rhs = EXPRESSION

NEGATIVE:  
allOf> entry = EXPRESSION

OF:  
allOf> operator = EXPRESSION, arguments = EXPRESSION

OTHERINTERVAL:  
someOf> lowerclosed = EXPRESSION, loweropen = EXPRESSION  
someOf> upperopen = EXPRESSION, upperclosed = EXPRESSION

PARTIAL:  
someOfType> Integer = EXPRESSION

PARENTHESIS:  
allOf> entry = NODE

PRIME:  
allOf> entry = EXPRESSION

POWER:  
allOf> base = EXPRESSION, exponent = EXPRESSION

PROB:  
allOf> event = NODE  
optional> condition = GEN\_STATEMENT

PLUS:  
someOfType> Integer = EXPRESSION

ROOT:  
allOf> entry = EXPRESSION  
optional> degree = EXPRESSION

SQRT:  
allOf> entry = EXPRESSION

SET2EXP:  
allOf> lhs = EXPRESSION, rhs = EXPRESSION, BINDS = VAR

SUM:

allOf> formula = EXPRESSION, BINDS = VARLIST  
optional> index = EXPRESSION, from = EXPRESSION, to = EXPRESSION

SUBSET:  
allOf> lhs = EXPRESSION, rhs = EXPRESSION

SUPERSET:  
allOf> lhs = EXPRESSION, rhs = EXPRESSION

SETMINUS:  
allOf> lhs = EXPRESSION, rhs = EXPRESSION

TIMES:  
someOfType> Integer = EXPRESSION

UNARYRELATION:  
allOf> entry = EXPRESSION

UNION:  
someOfType> Integer = EXPRESSION

VARLIST:  
someOfType> Integer = VAR

VECTOR:  
someOfType> Integer = EXPRESSION

SETBUCKET:  
someOfType> Integer = OBJOREXP

INTERVAL:  
allOf> lower = EXPRESSION, upper = EXPRESSION

INT:  
allOf> formula = EXPRESSION, variable = VAR, BINDS = VAR  
optional> index = EXPRESSION, from = EXPRESSION, to = EXPRESSION

FORMULA:  
allOf> entry = EXPRESSION

EMPTY:

SET:  
allOf> scopedvar = EXPRESSION, BINDS = VAR, condition = EXPRESSION

ORDEREDPAIR:  
allOf> first = EXPRESSION, second = EXPRESSION

RELATION:  
allOf> lhs = EXPRESSION, relation = NODE, rhs = EXPRESSION  
optional> above = EXPRESSION, restriction = RESTRICTION

LIST:  
someOfType> Integer = NODE

optional> separator = NODE, bracket = NODE

EQUAL:

IN:

LEQ:

LESS:

NOTEQUAL:

GEQ:

GREATER:

BINARY:

## 5 Not yet implemented

The following expressions involve some types that are not implemented yet:

1. Ellipsis for operations, e.g.,

$$\int \dots \int f(x_1, \dots, x_n) dx_1 \dots dx_n$$

2. Diagrams

3. Tables

4.  $\int \frac{x dx}{1+x^2}$  instead of  $\int \frac{x}{1+x^2} dx$

5. Other integral styles (Lebesgue etc.).

6.  $m(a) \begin{cases} > 0 & \text{if } A \in \mathcal{A} \\ = 0 & \text{otherwise} \end{cases}$

## 6 Examples of expressions

We give examples for the representation of expressions in the SM as records.

The following table gives a small statistic of the examples:

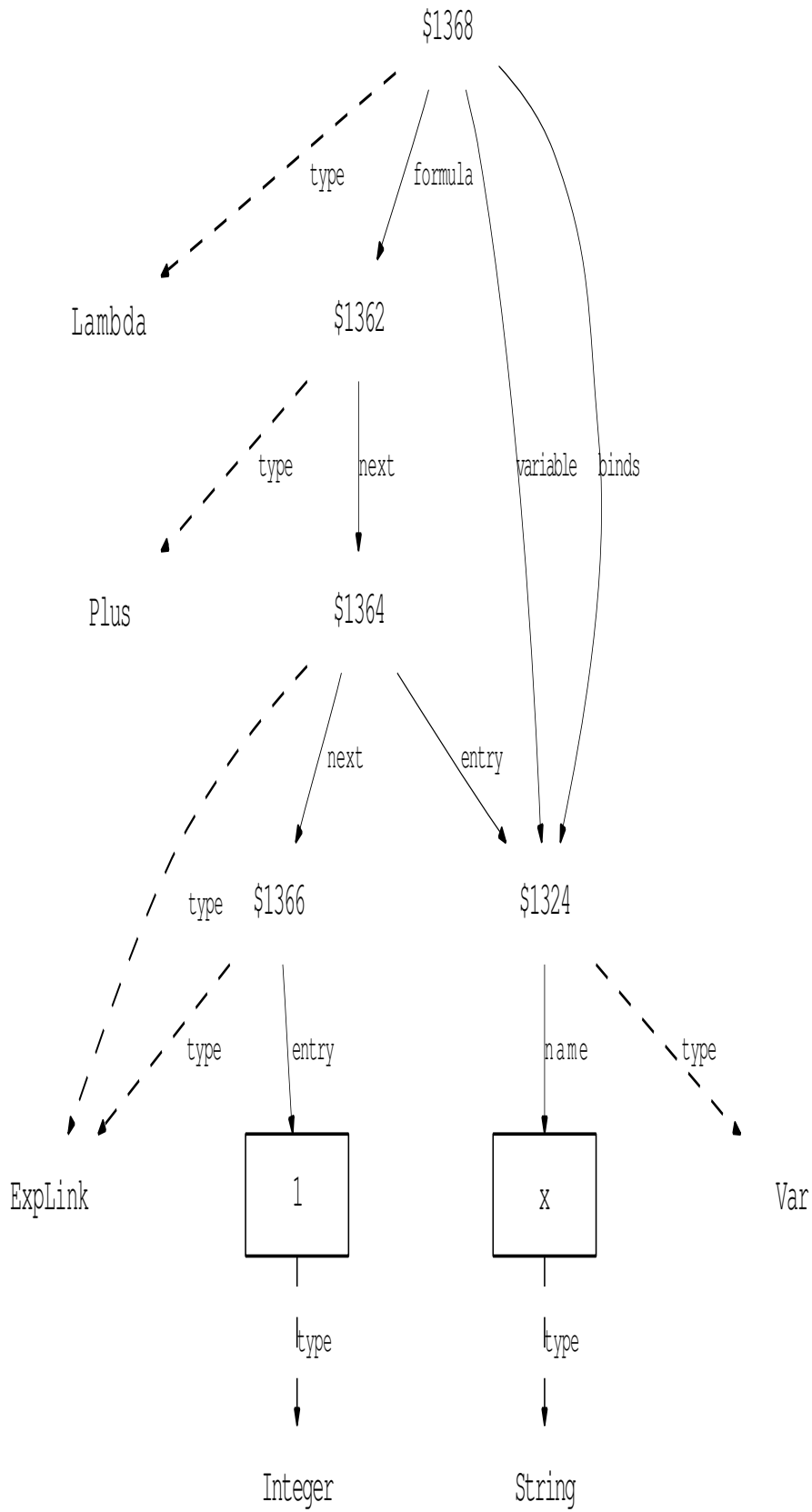
Example	# visible symbols	# L <sup>A</sup> T <sub>E</sub> X-characters	# sems
1	6	24	15
2	10	45	28
3	14	77	34
4	22	45	53
5	15	63	40
6	7	36	13
7	2	14	13
8	11	92	48
9	24	105	31
10	18	119	41
11	7	33	13
12	11	60	36
13	10	42	22
14	6	114	24
15	33	55	76
16	14	49	33
17	17	60	41
18	20	68	60
19	16	49	27
20	22	85	81
21	9	28	39
22	6	20	16
23	11	51	48
24	28	100	53
25	17	30	22
26	19	95	44
27	18	77	76
28	13	38	29
29	15	57	40
30	17	48	37
31	16	38	36

In the following examples, the entry of `#o.free` containing the free variables of the expression, is missing!

**Example 1.**

$$\lambda x.x + 1$$

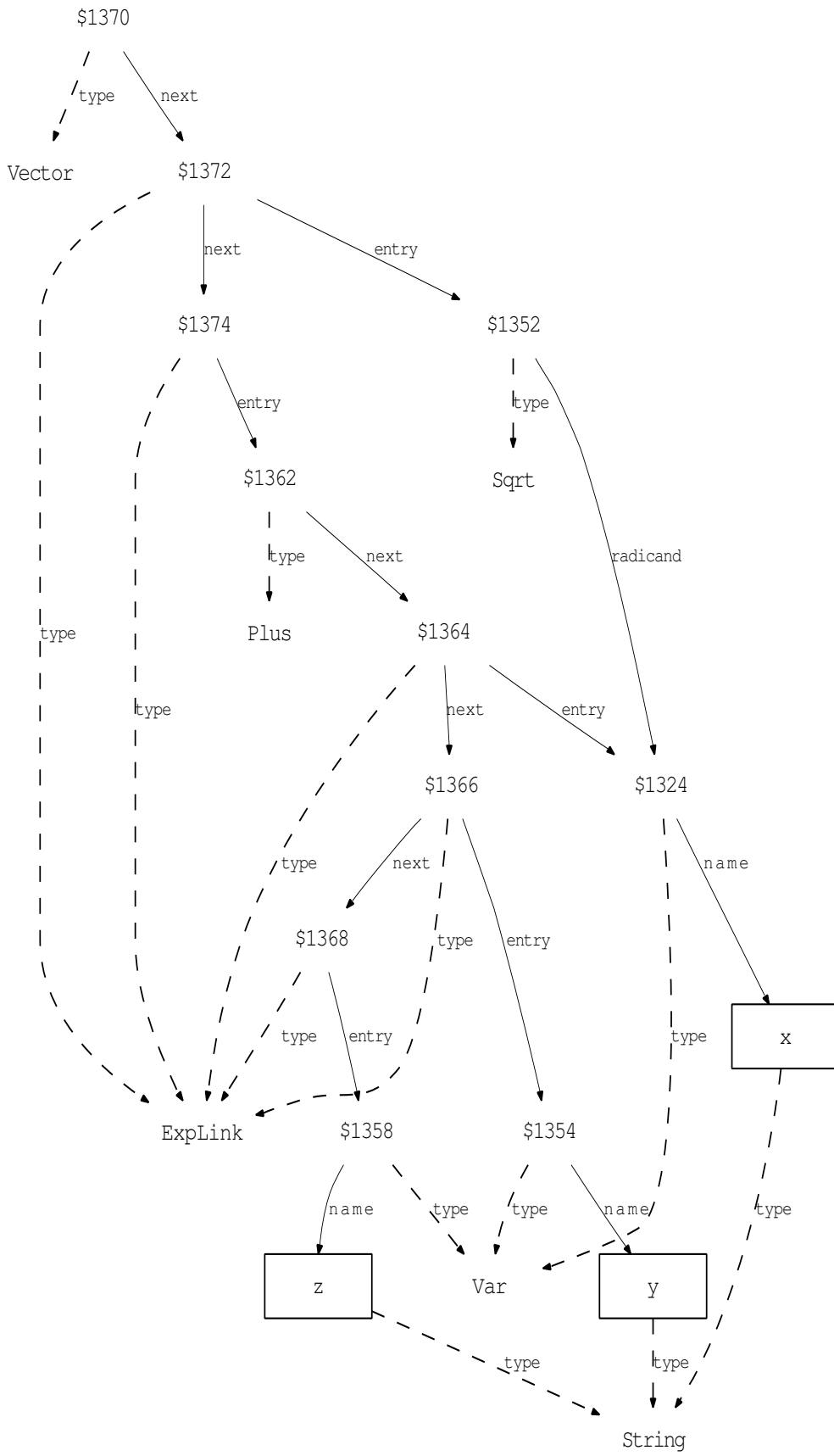
record	MATLAB construction code
\$1368.type=Lambda	x=createvar('x');
\$1368.formula=\$1362	one=createint('1');
\$1368.binds=\$1324	sum=create('Plus',{x,one});
\$1368.variable=\$1324	ex=create('Lambda',sum,x,x);
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1362.type=Plus	
\$1362.next=\$1364	
\$1364.type=ExpLink	
\$1364.next=\$1366	
\$1364.entry=\$1324	
\$1366.type=ExpLink	
\$1366.entry=\$1360	
\$1360.type=Integer	
VALUE(\$1342) = x	
VALUE(\$1360) = 1	



**Example 2.**

$$(\sqrt{x}, x + y + z)$$

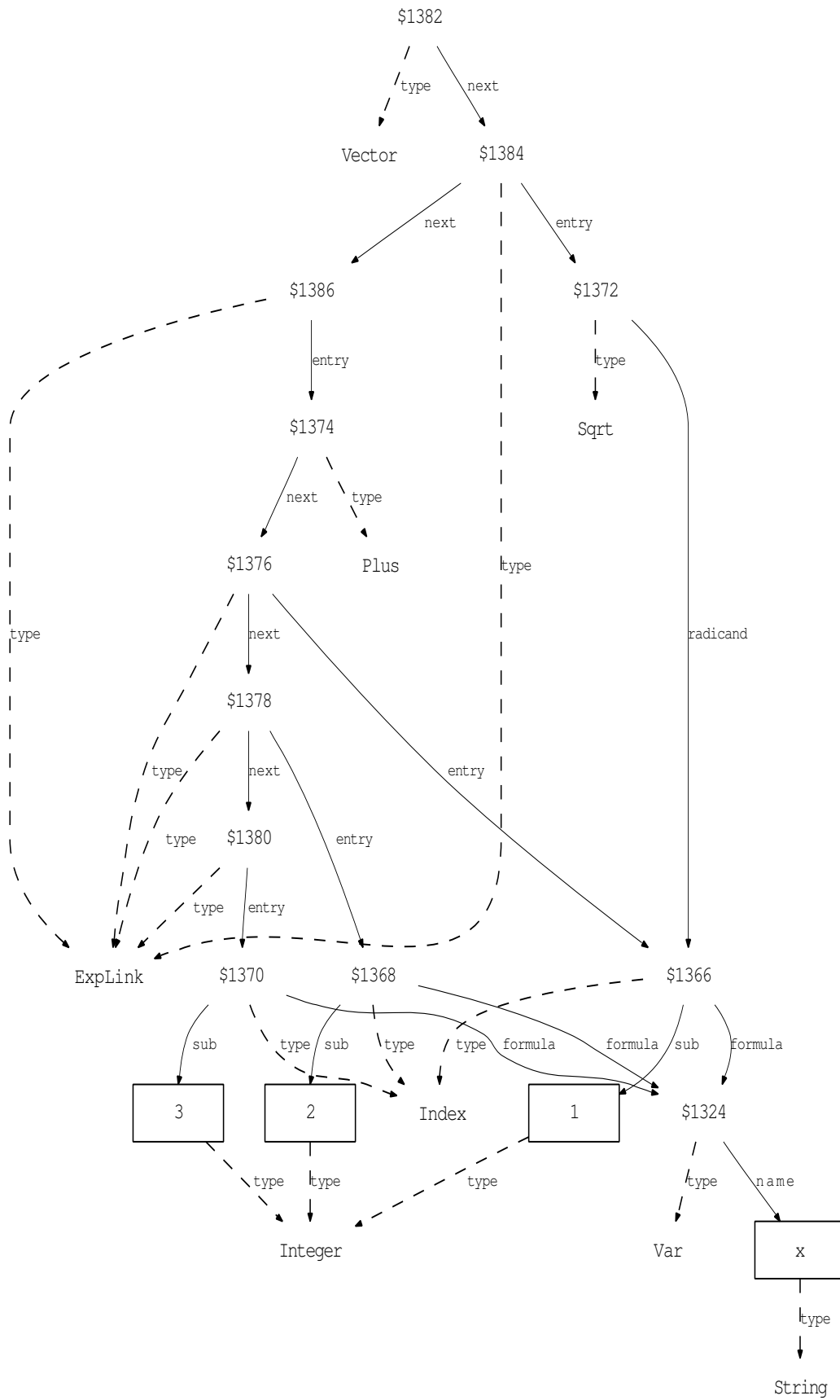
record	MATLAB construction code
\$1370.type=Vector	x=createvar('x');
\$1370.next=\$1372	sq=create('Sqrt',x);
\$1372.type=ExpLink	y=createvar('y');
\$1372.next=\$1374	z=createvar('z');
\$1372.entry=\$1352	sm=create('Plus',{x,y,z});
\$1352.type=Sqrt	ex=create('Vector',{sq,sm});
\$1352.radicand=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1374.type=ExpLink	
\$1374.entry=\$1362	
\$1362.type=Plus	
\$1362.next=\$1364	
\$1364.type=ExpLink	
\$1364.next=\$1366	
\$1364.entry=\$1324	
\$1366.type=ExpLink	
\$1366.next=\$1368	
\$1366.entry=\$1354	
\$1354.type=Var	
\$1354.name=\$1356	
\$1356.type=String	
\$1368.type=ExpLink	
\$1368.entry=\$1358	
\$1358.type=Var	
\$1358.name=\$1360	
\$1360.type=String	
VALUE(\$1342) = x	
VALUE(\$1356) = y	
VALUE(\$1360) = z	



**Example 3.** Similar as before, but now the only variable is the vector  $x$ .

$$(\sqrt{x_1}, x_1 + x_2 + x_3)$$

record	MATLAB construction code
\$1382.type=Vector	x=createvar('x');
\$1382.next=\$1384	one=createint('1');
\$1384.type=ExpLink	two=createint('2');
\$1384.next=\$1386	three=createint('3');
\$1384.entry=\$1372	x1=create('Index',x,one);
\$1372.type=Sqrt	x2=create('Index',x,two);
\$1372.radicand=\$1366	x3=create('Index',x,three);
\$1366.type=Index	sq=create('Sqrt',x1);
\$1366.formula=\$1324	sm=create('Plus',{x1,x2,x3});
\$1366.sub=\$1360	ex=create('Vector',{sq,sm});
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1360.type=Integer	
\$1386.type=ExpLink	
\$1386.entry=\$1374	
\$1374.type=Plus	
\$1374.next=\$1376	
\$1376.type=ExpLink	
\$1376.next=\$1378	
\$1376.entry=\$1366	
\$1378.type=ExpLink	
\$1378.next=\$1380	
\$1378.entry=\$1368	
\$1368.type=Index	
\$1368.formula=\$1324	
\$1368.sub=\$1362	
\$1362.type=Integer	
\$1380.type=ExpLink	
\$1380.entry=\$1370	
\$1370.type=Index	
\$1370.formula=\$1324	
\$1370.sub=\$1364	
\$1364.type=Integer	
VALUE(\$1342) = x	
VALUE(\$1360) = 1	
VALUE(\$1362) = 2	
VALUE(\$1364) = 3	



**Example 4.** The variable  $y$  is the only free variable.

$$\forall x, z \in X : f(x, y, z) = g(y, x)$$

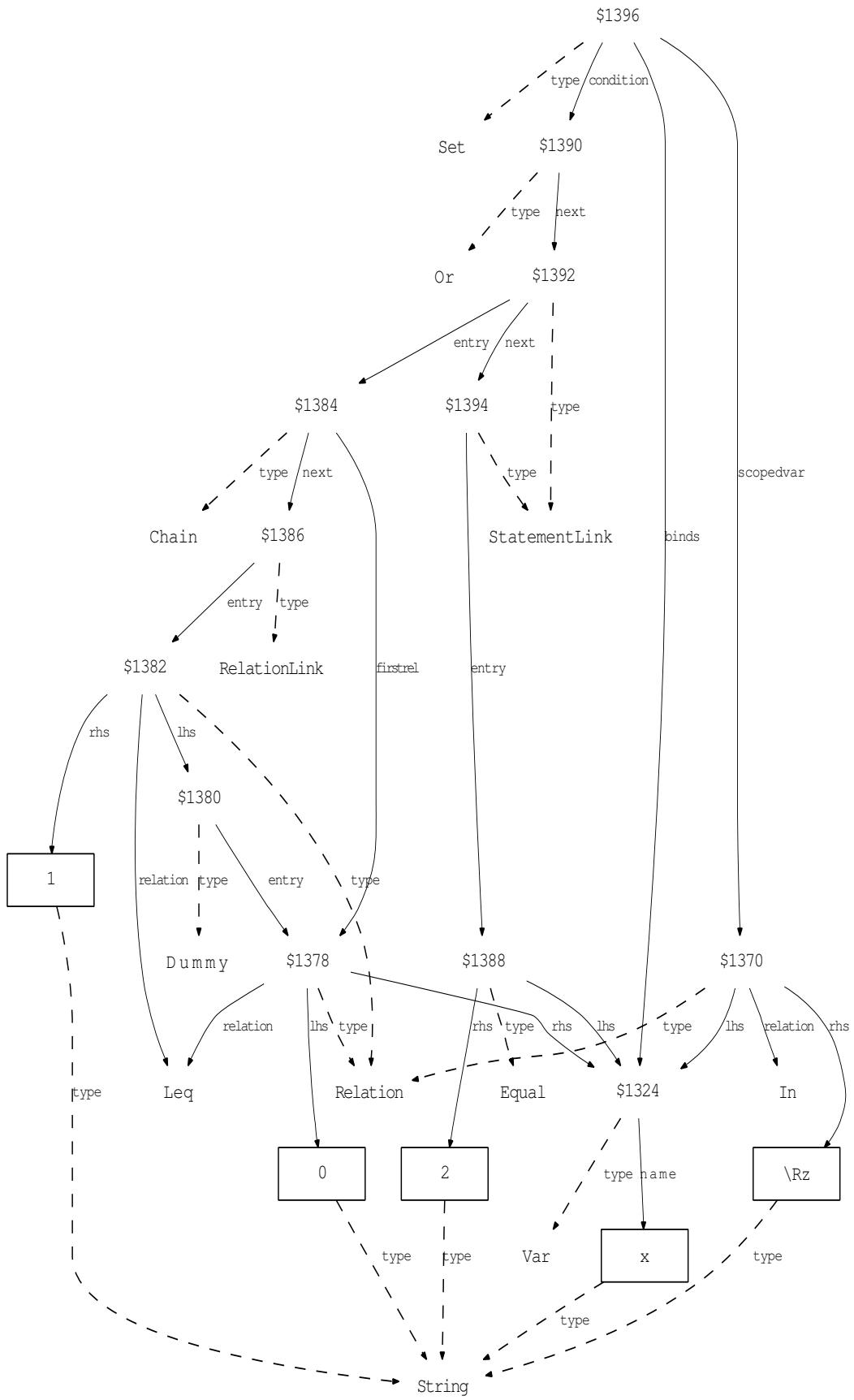
record	MATLAB construction code
\$1404.type=Forall	x=createvar('x');
\$1404.formula=\$1384	y=createvar('y');
\$1404.binds=\$1386	z=createvar('z');
\$1404.scopedvar=\$1402	f=createstring('f');
\$1384.type=Equal	g=createstring('g');
\$1384.lhs=\$1380	X=createstring('X');
\$1384.rhs=\$1382	xyz=create('List',{x,y,z});
\$1380.type=Of	yx=create('List',{y,x});
\$1380.operator=\$1360	lhs=create('Of',f,xyz);
\$1380.arguments=\$1366	rhs=create('Of',g,yx);
\$1360.type=String	eq=create('Equal',lhs,rhs);
\$1366.type=List	xz = create('VarList',{x,z});
\$1366.next=\$1368	in=createname('in');
\$1368.type=ObjLink	dummy=create('Dummy',x);
\$1368.next=\$1370	var=create('Relation',xz,in,X);
\$1368.entry=\$1324	ex=create('Forall',eq,var,xz);
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1370.type=ObjLink	
\$1370.next=\$1372	
\$1370.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1372.type=ObjLink	
\$1372.entry=\$1356	
\$1356.type=Var	
\$1356.name=\$1358	
\$1358.type=String	
\$1382.type=Of	
\$1382.operator=\$1362	
\$1382.arguments=\$1374	
\$1362.type=String	
\$1374.type=List	
\$1374.next=\$1376	
\$1376.type=ObjLink	
\$1376.next=\$1378	
\$1376.entry=\$1352	
\$1378.type=ObjLink	
\$1378.entry=\$1324	
\$1386.type=VarList	
\$1386.next=\$1388	
\$1388.type=VarLink	
\$1388.next=\$1390	
\$1388.entry=\$1324	
\$1390.type=VarLink	
\$1390.entry=\$1356	
\$1402.type=Relation	
\$1402.lhs=\$1386	
\$1402.rhs=\$1364	
\$1402.relation=in	
\$1364.type=String	
VALUE(\$1342) = x	
VALUE(\$1354) = y	
VALUE(\$1358) = z	
VALUE(\$1360) = f	
VALUE(\$1362) = g	
VALUE(\$1364) = X	



Example 5.

$$\{x \in \mathbb{R} \mid 0 \leq x \leq 1 \vee x=2\}$$

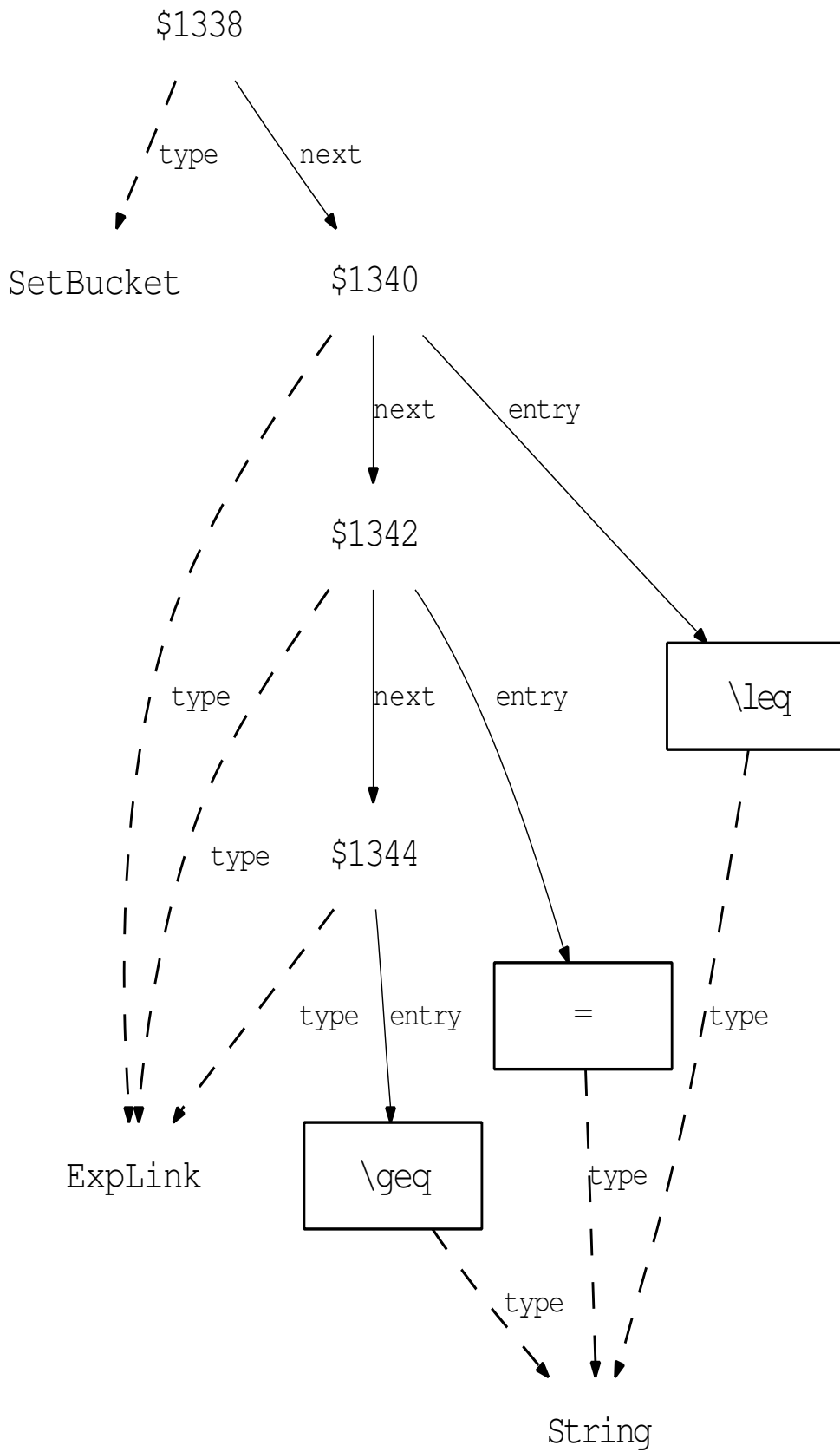
record	MATLAB construction code
\$1396.type=Set	x=createvar('x');
\$1396.condition=\$1390	R=createstring('\Rz');
\$1396.binds=\$1324	in=createnam('In');
\$1396.scopedvar=\$1370	leq=createnam('Leq');
\$1324.type=Var	xinR=create('Relation',x,in,R);
\$1324.name=\$1342	zero=createstring('0');
\$1342.type=String	one=createstring('1');
\$1370.type=Relation	two=createstring('2');
\$1370.lhs=\$1324	f1a=create('Relation',zero,leq,x);
\$1370.rhs=\$1352	dummy=create('Dummy',f1a);
\$1370.relation=In	f1b=create('Relation',dummy,leq,one);
\$1352.type=String	f1=create('Chain',f1a,{f1b});
\$1390.type=Or	f2=create('Equal',x,two);
\$1390.next=\$1392	exor=create('Or',{f1,f2});
\$1392.type=StatementLink	ex=create('Set',x,xinR,exor);
\$1392.next=\$1394	
\$1392.entry=\$1384	
\$1384.type=Chain	
\$1384.next=\$1386	
\$1384.firstrel=\$1378	
\$1378.type=Relation	
\$1378.lhs=\$1372	
\$1378.rhs=\$1324	
\$1378.relation=Leq	
\$1372.type=String	
\$1386.type=RelationLink	
\$1386.entry=\$1382	
\$1382.type=Relation	
\$1382.lhs=\$1380	
\$1382.rhs=\$1374	
\$1382.relation=Leq	
\$1374.type=String	
\$1380.type=Dummy	
\$1380.entry=\$1378	
\$1394.type=StatementLink	
\$1394.entry=\$1388	
\$1388.type=Equal	
\$1388.lhs=\$1324	
\$1388.rhs=\$1376	
\$1376.type=String	
VALUE(\$1342) = x	
VALUE(\$1352) = \Rz	
VALUE(\$1372) = 0	
VALUE(\$1374) = 1	
VALUE(\$1376) = 2	



**Example 6.** A set as list with characters as entries.

$$\{\leq, =, \geq\}$$

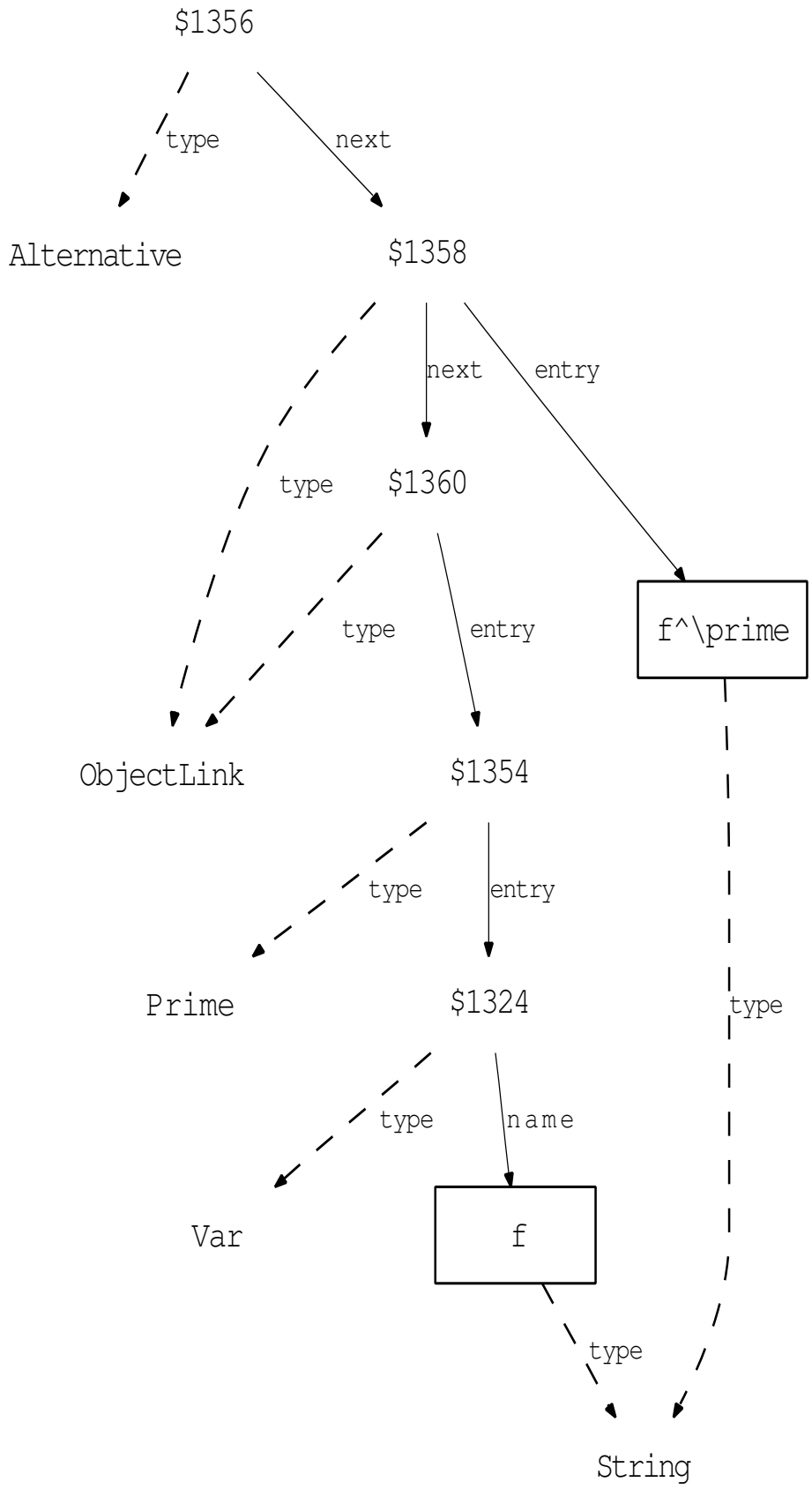
record	MATLAB construction code
\$1338.type=SetBucket	le=createstring('\leq');
\$1338.next=\$1340	eq=createstring('=');
\$1340.type=ExpLink	ge=createstring('\geq');
\$1340.next=\$1342	ex=create('SetBucket',{le,eq,ge});
\$1340.entry=\$1332	
\$1332.type=String	
\$1342.type=ExpLink	
\$1342.next=\$1344	
\$1342.entry=\$1334	
\$1334.type=String	
\$1344.type=ExpLink	
\$1344.entry=\$1336	
\$1336.type=String	
VALUE(\$1332) = \leq	
VALUE(\$1334) = =	
VALUE(\$1336) = \geq	



**Example 7.**  $f'$  is ambiguous: It could be the application of the operation  $'$  to the mapping  $f$  or the name of fuzzy numbers  $f, f', f''$ .

$f'$

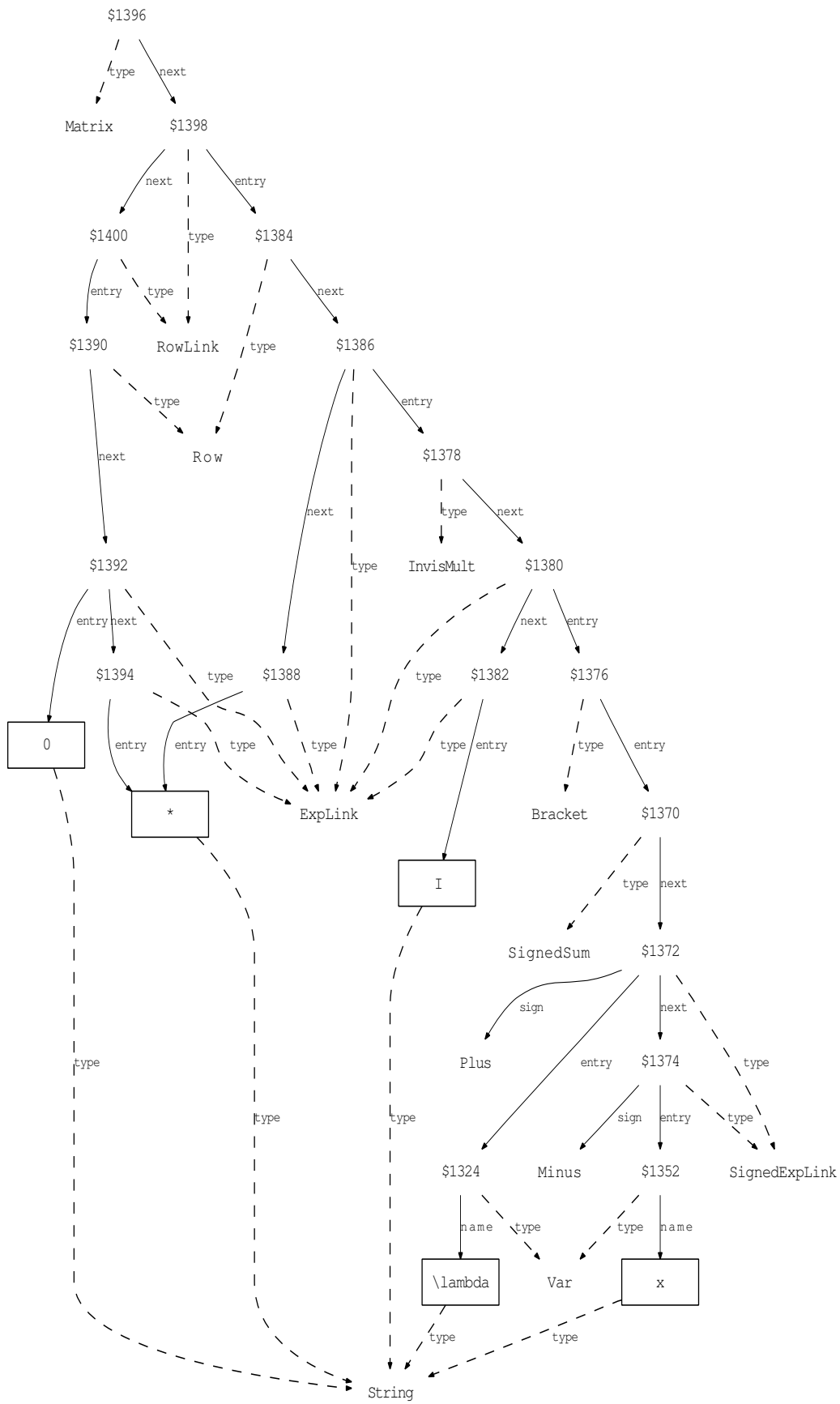
record	MATLAB construction code
\$1356.type=Alternative	f=createvar('f');
\$1356.next=\$1358	fpr1=createstring('f^\prime');
\$1358.type=ObjectLink	fpr2=create('Prime',f);
\$1358.next=\$1360	ex=create('Alternative',{fpr1,fpr2});
\$1358.entry=\$1352	
\$1352.type=String	
\$1360.type=ObjectLink	
\$1360.entry=\$1354	
\$1354.type=Prime	
\$1354.entry=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
VALUE(\$1342) = f	
VALUE(\$1352) = f^\prime	



**Example 8.** A matrix with wildcard characters, denoted by \*.

$$\begin{pmatrix} (\lambda - x)I & * \\ 0 & * \end{pmatrix}$$

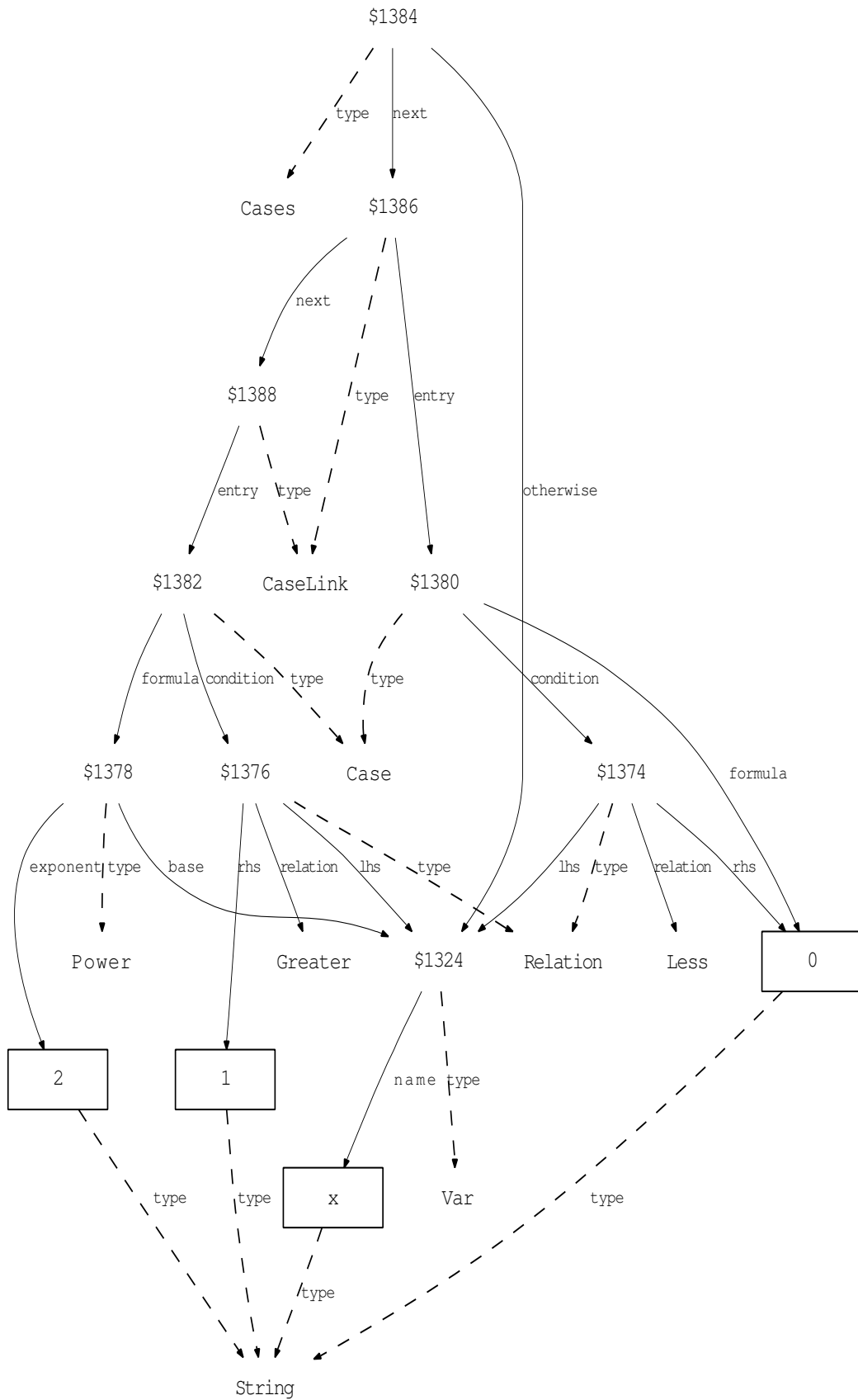
record	MATLAB construction code
\$1396.type=Matrix	lambda=createvar('\lambda');
\$1396.next=\$1398	x=createvar('x');
\$1398.type=RowLink	id=createstring('I');
\$1398.next=\$1400	ast=createstring('*');
\$1398.entry=\$1384	zero=createstring('0');
\$1384.type=Row	plus=create('Plus');
\$1384.next=\$1386	minus=create('Minus');
\$1386.type=ExpLink	mi=create('SignedSum',{plus,lambda},{minus,x});
\$1386.next=\$1388	bra=create('Bracket',mi);
\$1386.entry=\$1378	mult=create('InvisMult',{bra,id});
\$1378.type=InvisMult	row1=create('Row',{mult,ast});
\$1378.next=\$1380	row2=create('Row',{zero,ast});
\$1380.type=ExpLink	ex=create('Matrix',{row1,row2});
\$1380.next=\$1382	
\$1380.entry=\$1376	
\$1376.type=Bracket	
\$1376.entry=\$1370	
\$1370.type=SignedSum	
\$1370.next=\$1372	
\$1372.type=SignedExpLink	
\$1372.next=\$1374	
\$1372.entry=\$1324	
\$1372.sign=Plus	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1374.type=SignedExpLink	
\$1374.entry=\$1352	
\$1374.sign=Minus	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1382.type=ExpLink	
\$1382.entry=\$1356	
\$1356.type=String	
\$1388.type=ExpLink	
\$1388.entry=\$1358	
\$1358.type=String	
\$1400.type=RowLink	
\$1400.entry=\$1390	
\$1390.type=Row	
\$1390.next=\$1392	
\$1392.type=ExpLink	
\$1392.next=\$1394	
\$1392.entry=\$1360	
\$1360.type=String	
\$1394.type=ExpLink	
\$1394.entry=\$1358	
VALUE(\$1342) = \lambda	
VALUE(\$1354) = x	
VALUE(\$1356) = I	
VALUE(\$1358) = *	
VALUE(\$1360) = 0	



**Example 9.**

$$\begin{cases} 0 & \text{if } x < 0 \\ x^2 & \text{if } x > 1 \\ x & \text{otherwise} \end{cases}$$

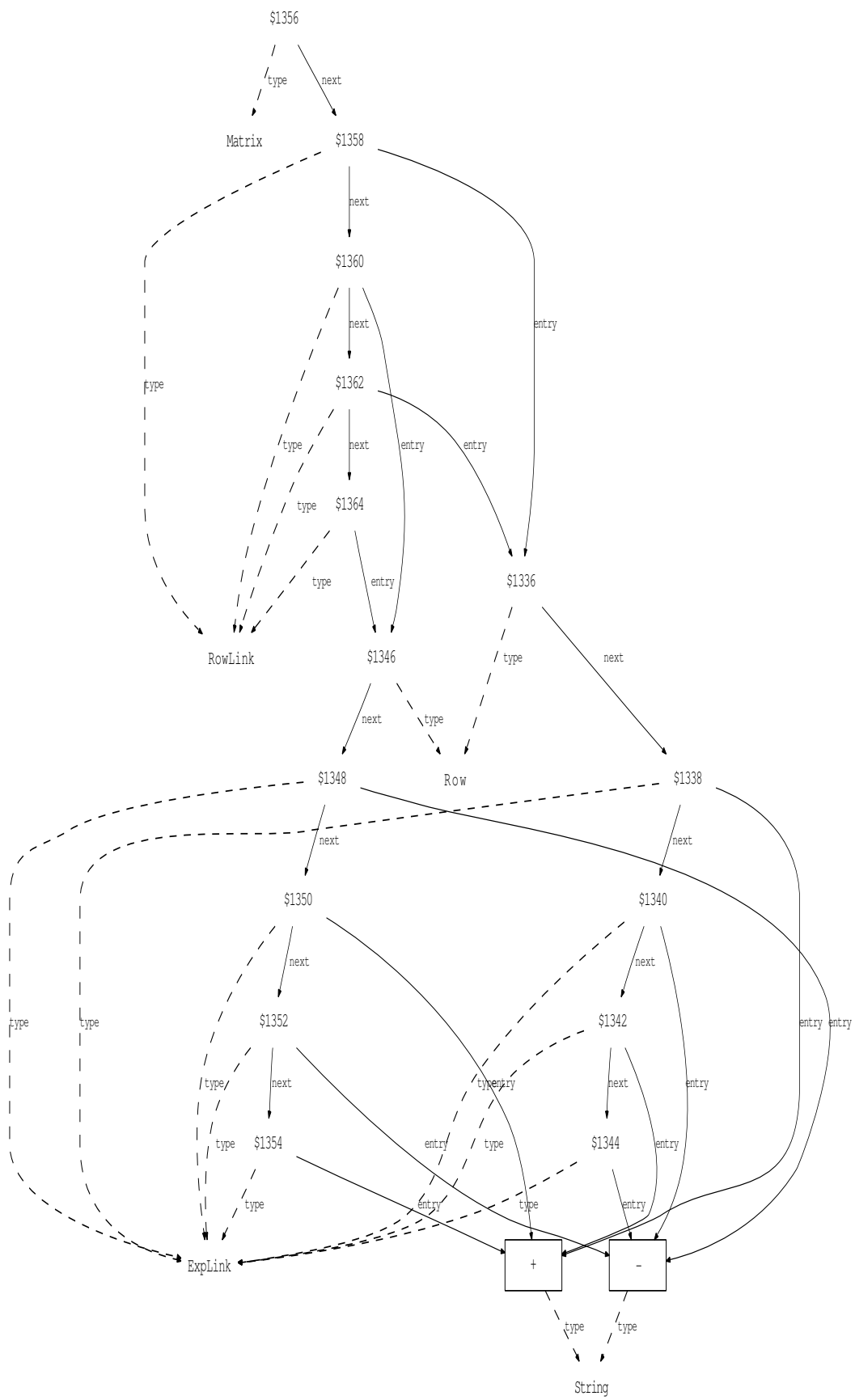
record	MATLAB construction code
\$1384.type=Cases	x=createvar('x');
\$1384.next=\$1386	zero=createstring('0');
\$1384.otherwise=\$1324	one=createstring('1');
\$1324.type=Var	two=createstring('2');
\$1324.name=\$1342	less = createname('Less');
\$1342.type=String	greater = createname('Greater');
\$1386.type=CaseLink	le=create('Relation',x,less,zero);
\$1386.next=\$1388	gr=create('Relation',x,greater,one);
\$1386.entry=\$1380	sq=create('Power',x,two);
\$1380.type=Case	c1=create('Case',zero,le);
\$1380.formula=\$1352	c2=create('Case',sq,gr);
\$1380.condition=\$1374	ex=create('Cases',{c1,c2},x);
\$1352.type=String	
\$1374.type=Relation	
\$1374.lhs=\$1324	
\$1374.rhs=\$1352	
\$1374.relation=Less	
\$1388.type=CaseLink	
\$1388.entry=\$1382	
\$1382.type=Case	
\$1382.formula=\$1378	
\$1382.condition=\$1376	
\$1376.type=Relation	
\$1376.lhs=\$1324	
\$1376.rhs=\$1354	
\$1376.relation=Greater	
\$1354.type=String	
\$1378.type=Power	
\$1378.base=\$1324	
\$1378.exponent=\$1356	
\$1356.type=String	
VALUE(\$1342) = x	
VALUE(\$1352) = 0	
VALUE(\$1354) = 1	
VALUE(\$1356) = 2	



**Example 10.**

$$\begin{pmatrix} + & - & + & - \\ - & + & - & + \\ + & - & + & - \\ - & + & - & + \end{pmatrix}$$

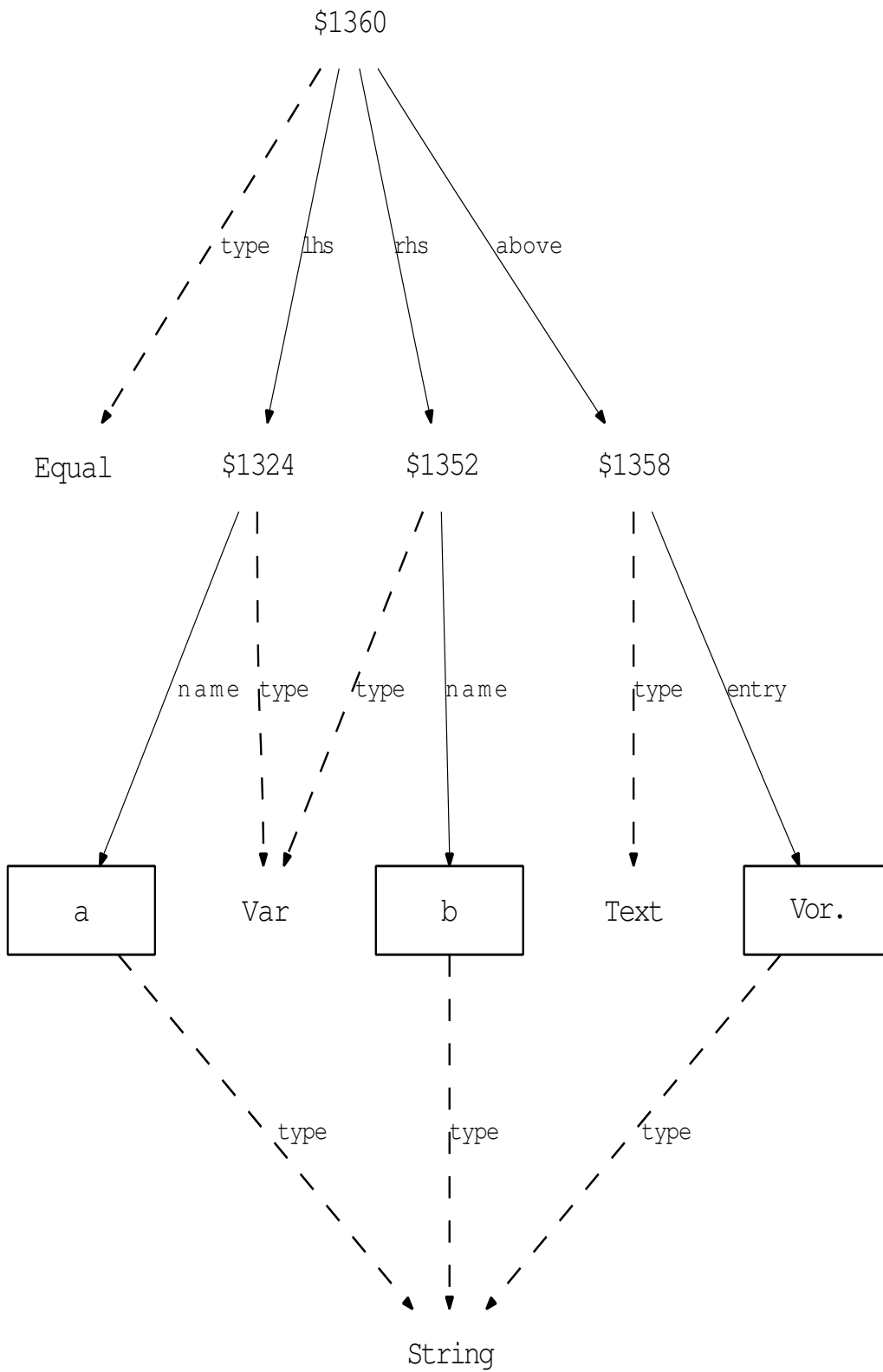
record	MATLAB construction code
\$1356.type=Matrix	pl=createstring('+');
\$1356.next=\$1358	mi=createstring('-');
\$1358.type=RowLink	row1=create('Row',{pl,mi,pl,mi});
\$1358.next=\$1360	row2=create('Row',{mi,pl,mi,pl});
\$1358.entry=\$1336	ex=create('Matrix',{row1,row2,row1,row2});
\$1336.type=Row	
\$1336.next=\$1338	
\$1338.type=Explink	
\$1338.next=\$1340	
\$1338.entry=\$1332	
\$1332.type=String	
\$1340.type=Explink	
\$1340.next=\$1342	
\$1340.entry=\$1334	
\$1334.type=String	
\$1342.type=Explink	
\$1342.next=\$1344	
\$1342.entry=\$1332	
\$1344.type=Explink	
\$1344.entry=\$1334	
\$1360.type=RowLink	
\$1360.next=\$1362	
\$1360.entry=\$1346	
\$1346.type=Row	
\$1346.next=\$1348	
\$1348.type=Explink	
\$1348.next=\$1350	
\$1348.entry=\$1334	
\$1350.type=Explink	
\$1350.next=\$1352	
\$1350.entry=\$1332	
\$1352.type=Explink	
\$1352.next=\$1354	
\$1352.entry=\$1334	
\$1354.type=Explink	
\$1354.entry=\$1332	
\$1362.type=RowLink	
\$1362.next=\$1364	
\$1362.entry=\$1336	
\$1364.type=RowLink	
\$1364.entry=\$1346	
VALUE(\$1332) = +	
VALUE(\$1334) = -	



**Example 11.** Equality with auxiliary information, stored in the node above.

$$a \stackrel{\text{Vor.}}{=} b$$

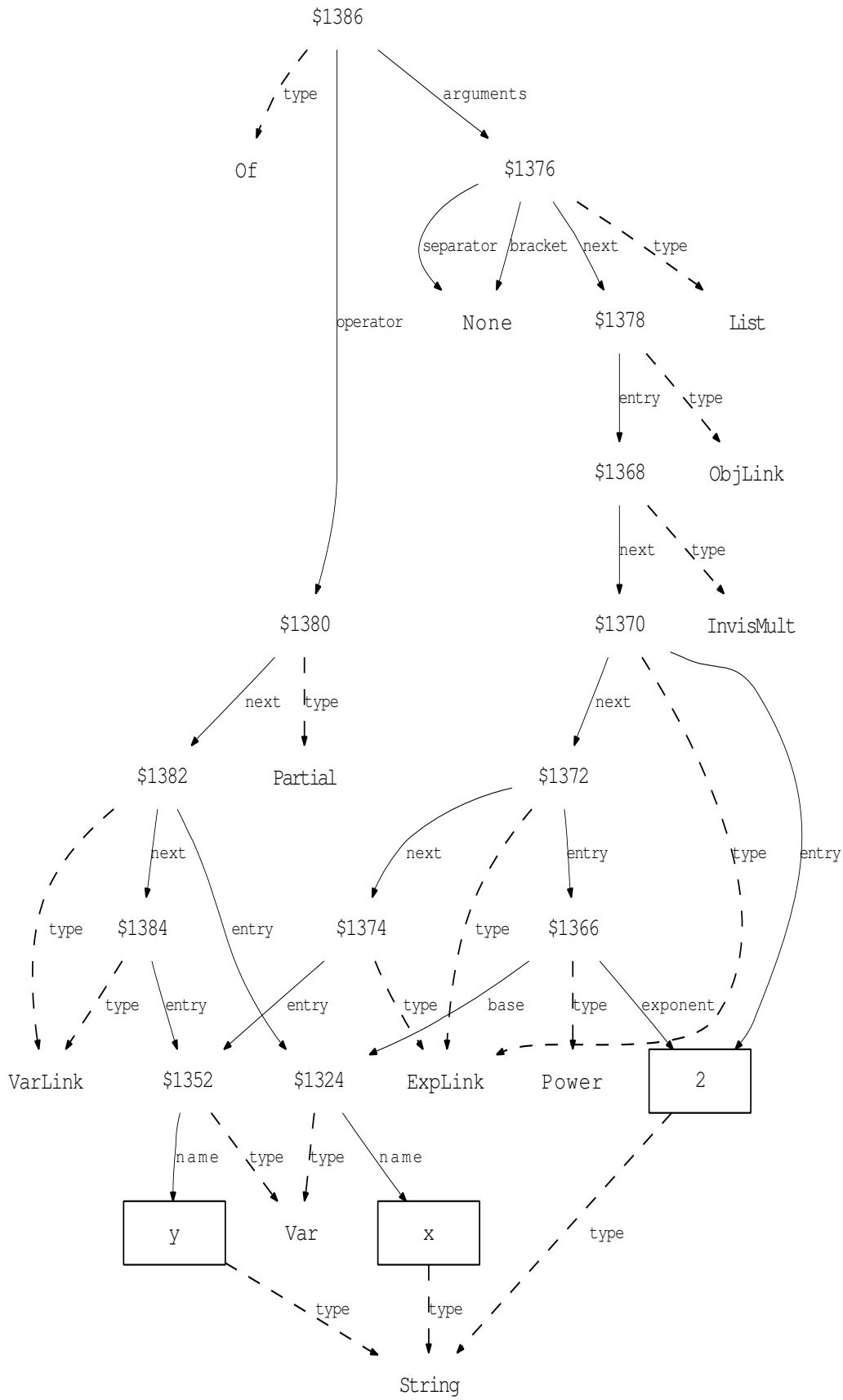
record	MATLAB construction code
\$1360.type=Equal	a=createvar('a');
\$1360.lhs=\$1324	b=createvar('b');
\$1360.rhs=\$1352	vorstr=createstring('Vor.');
\$1360.above=\$1358	vor=create('Text',vorstr);
\$1324.type=Var	ex=create('Equal',a,b,vor);
\$1324.name=\$1342	
\$1342.type=String	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1358.type=Text	
\$1358.entry=\$1356	
\$1356.type=String	
VALUE(\$1342) = a	
VALUE(\$1354) = b	
VALUE(\$1356) = Vor.	



Example 12.

$$\frac{\partial^2}{\partial x \partial y} 2x^2y$$

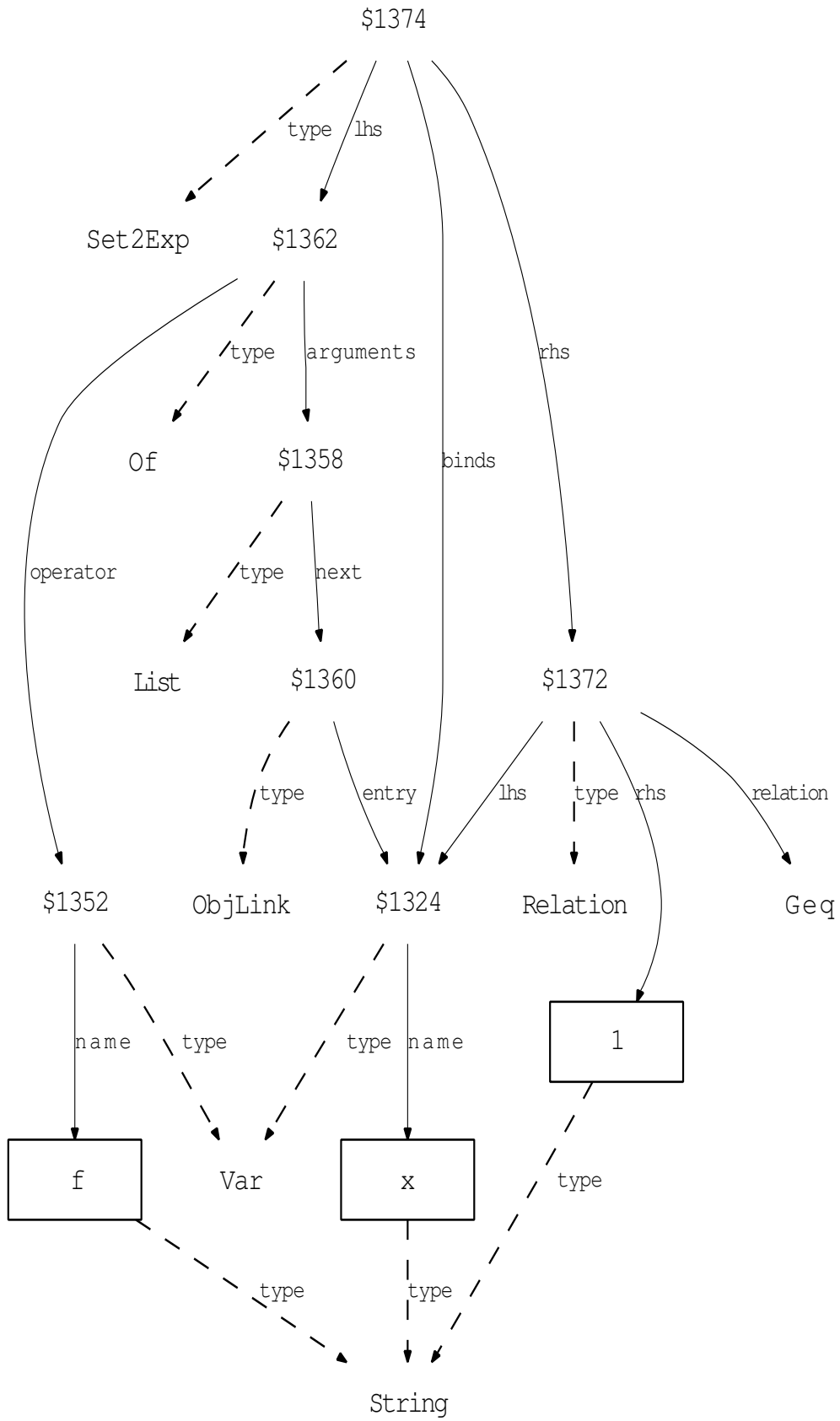
record	MATLAB construction code
\$1386.type=Of	x=createvar('x');
\$1386.operator=\$1380	y=createvar('y');
\$1386.arguments=\$1376	two=createstring('2');
\$1376.type=List	none=createname('None');
\$1376.next=\$1378	sq=create('Power',x,two);
\$1376.separator=None	mlt=create('InvisMult',{two,sq,y});
\$1376.bracket=None	args=create('List',{mlt},none,none);
\$1378.type=ObjLink	pder=create('Partial',{x,y});
\$1378.entry=\$1368	ex=create('Of',pder,args);
\$1368.type=InvisMult	
\$1368.next=\$1370	
\$1370.type=Explink	
\$1370.next=\$1372	
\$1370.entry=\$1356	
\$1356.type=String	
\$1372.type=Explink	
\$1372.next=\$1374	
\$1372.entry=\$1366	
\$1366.type=Power	
\$1366.base=\$1324	
\$1366.exponent=\$1356	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1374.type=Explink	
\$1374.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1380.type=Partial	
\$1380.next=\$1382	
\$1382.type=VarLink	
\$1382.next=\$1384	
\$1382.entry=\$1324	
\$1384.type=VarLink	
\$1384.entry=\$1352	
VALUE(\$1342) = x	
VALUE(\$1354) = y	
VALUE(\$1356) = 2	



**Example 13.** Another typical way of denoting a set: expressions left and right.

$$\{f(x) \mid x \geq 1\}$$

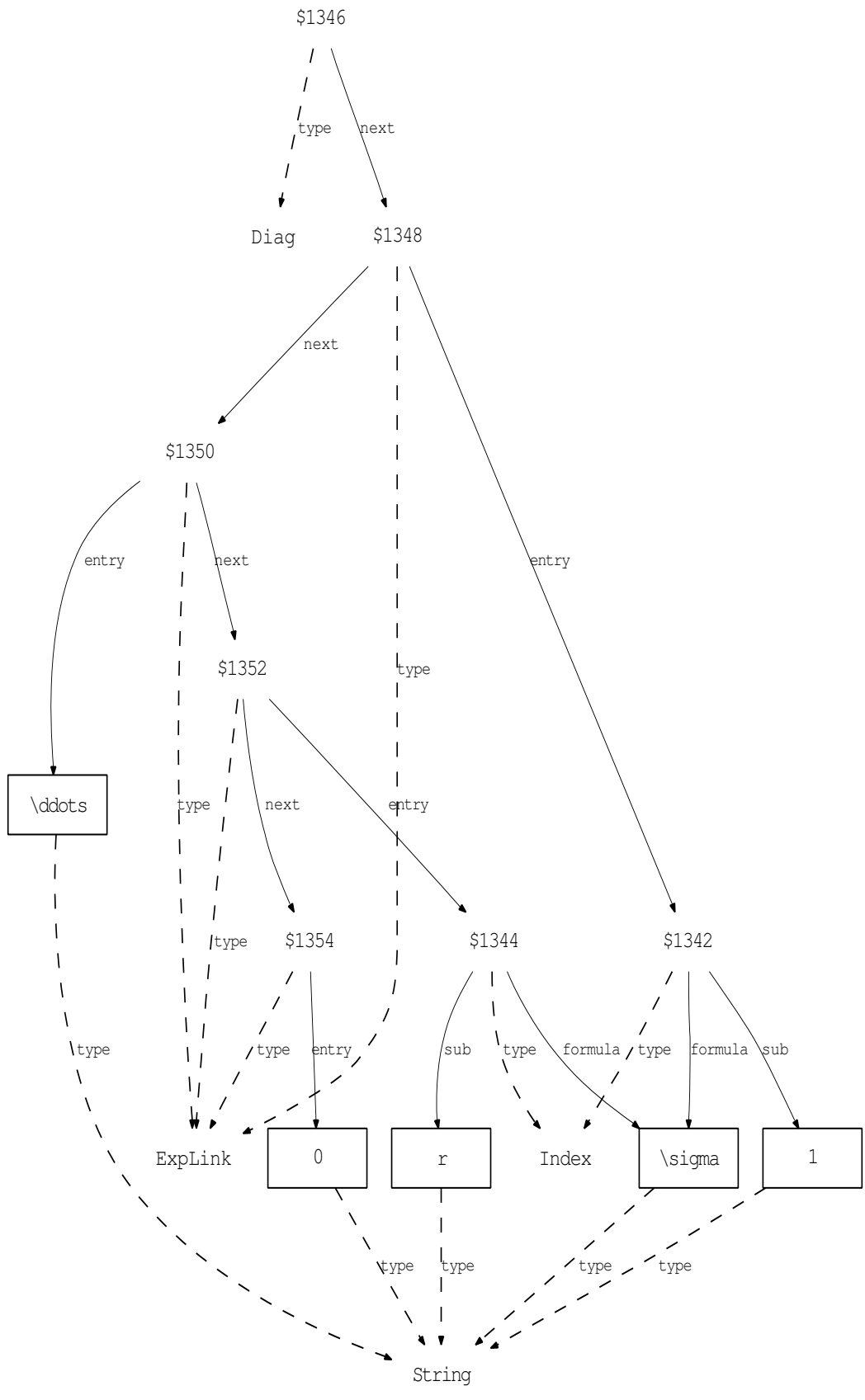
record	MATLAB construction code
\$1374.type=Set2Exp	x=createvar('x');
\$1374.lhs=\$1362	f=createvar('f');
\$1374.rhs=\$1372	one=createstring('1');
\$1374.binds=\$1324	args=create('List',{x});
\$1324.type=Var	fkt=create('Of',f,args);
\$1324.name=\$1342	geq=createname('Geq');
\$1342.type=String	ass=create('Relation',x,geq,one);
\$1362.type=Of	ex=create('Set2Exp',fkt,ass,x);
\$1362.operator=\$1352	
\$1362.arguments=\$1358	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1358.type=List	
\$1358.next=\$1360	
\$1360.type=ObjLink	
\$1360.entry=\$1324	
\$1372.type=Relation	
\$1372.lhs=\$1324	
\$1372.rhs=\$1356	
\$1372.relation=Geq	
\$1356.type=String	
VALUE(\$1342) = x	
VALUE(\$1354) = f	
VALUE(\$1356) = 1	



Example 14.

$$\begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & 0 \end{pmatrix}$$

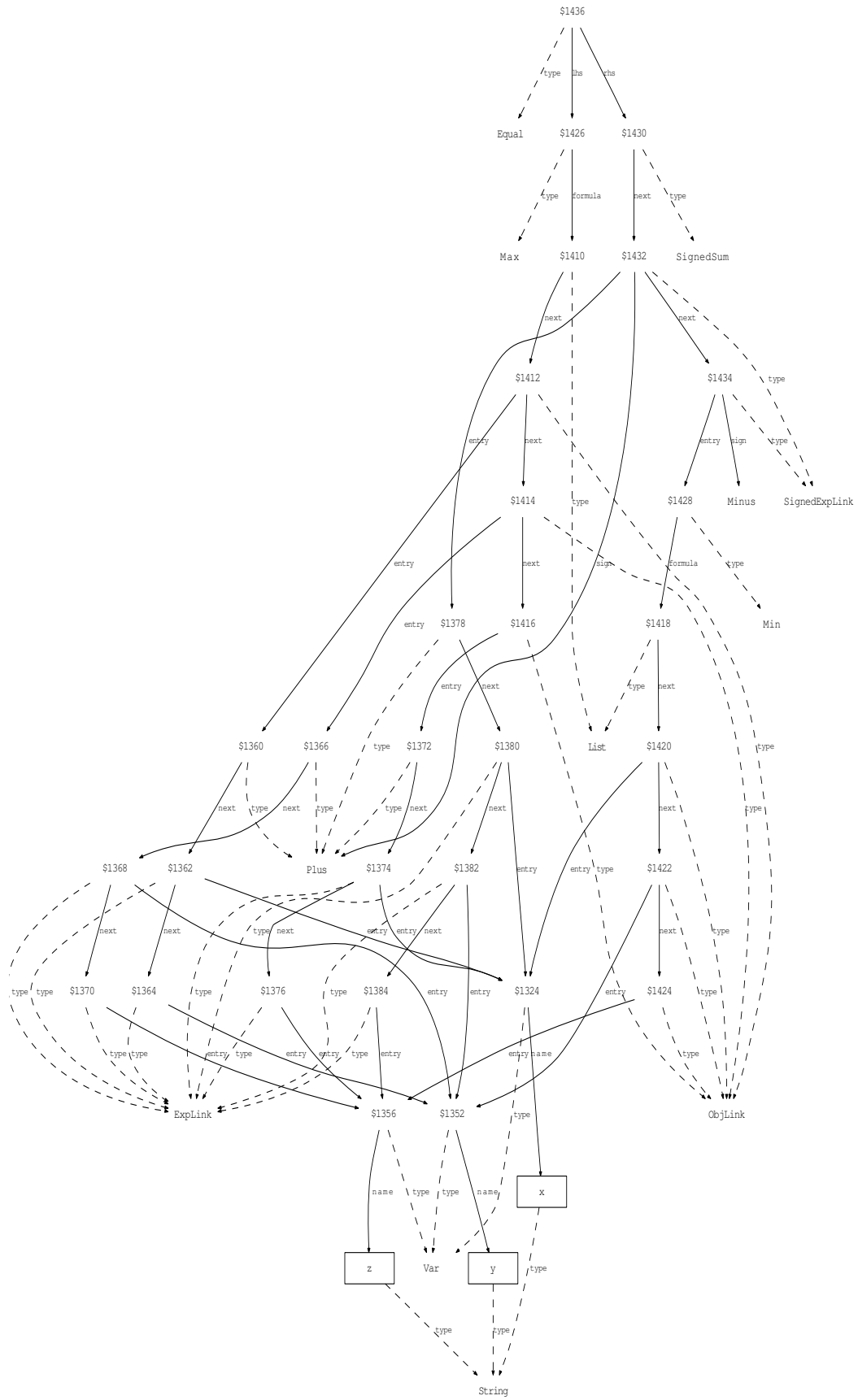
record	MATLAB construction code
\$1346.type=Diag	sig=createstring('\sigma');
\$1346.next=\$1348	one=createstring('1');
\$1348.type=ExpLink	r=createstring('r');
\$1348.next=\$1350	zero=createstring('0');
\$1348.entry=\$1342	dots=createstring('\ddots');
\$1342.type=Index	s1=create('Index',sig,one);
\$1342.formula=\$1332	sr=create('Index',sig,r);
\$1342.sub=\$1334	ex=create('Diag',{s1,dots,sr,zero});
\$1332.type=String	
\$1334.type=String	
\$1350.type=ExpLink	
\$1350.next=\$1352	
\$1350.entry=\$1340	
\$1340.type=String	
\$1352.type=ExpLink	
\$1352.next=\$1354	
\$1352.entry=\$1344	
\$1344.type=Index	
\$1344.formula=\$1332	
\$1344.sub=\$1336	
\$1336.type=String	
\$1354.type=ExpLink	
\$1354.entry=\$1338	
\$1338.type=String	
VALUE(\$1332) = \sigma	
VALUE(\$1334) = 1	
VALUE(\$1336) = r	
VALUE(\$1338) = 0	
VALUE(\$1340) = \ddots	



### Example 15.

$$\max(x + y, y + z, x + z) = x + y + z - \min(x, y, z)$$

record	MATLAB construction code
\$1436.type=Equal	x=createvar('x');
\$1436.lhs=\$1426	y=createvar('y');
\$1436.rhs=\$1430	z=createvar('z');
\$1426.type=Max	xy=create('Plus',{x,y});
\$1426.formula=\$1410	yz=create('Plus',{y,z});
\$1410.type=List	xz=create('Plus',{x,z});
\$1410.next=\$1412	xyz=create('Plus',{x,y,z});
\$1412.type=ObjLink	plus=createnamename('Plus');
\$1412.next=\$1414	minus=createnamename('Minus');
\$1412.entry=\$1360	none=createnamename('None');
\$1360.type=Plus	komma=createnamename('Komma');
\$1360.next=\$1362	l1=create('List',{xy,yz,xz});
\$1362.type=ExpLink	l2=create('List',{x,y,z});
\$1362.next=\$1364	max=create('Max',l1);
\$1362.entry=\$1324	min=create('Min',l2);
\$1324.type=Var	rhs=create('SignedSum',{plus,xyz},{minus,min});
\$1324.name=\$1342	ex=create('Equal',max,rhs);
\$1342.type=String	
\$1364.type=ExpLink	
\$1364.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1414.type=ObjLink	
\$1414.next=\$1416	
\$1414.entry=\$1366	
\$1366.type=Plus	
\$1366.next=\$1368	
\$1368.type=ExpLink	
\$1368.next=\$1370	
\$1368.entry=\$1352	
\$1370.type=ExpLink	
\$1370.entry=\$1356	
\$1356.type=Var	
\$1356.name=\$1358	
\$1358.type=String	
\$1416.type=ObjLink	
\$1416.entry=\$1372	
\$1372.type=Plus	
\$1372.next=\$1374	
\$1374.type=ExpLink	
\$1374.next=\$1376	
\$1374.entry=\$1324	
\$1376.type=ExpLink	
\$1376.entry=\$1356	
\$1430.type=SignedSum	
\$1430.next=\$1432	
\$1432.type=SignedExpLink	
\$1432.next=\$1434	
\$1432.entry=\$1378	
\$1432.sign=Plus	
\$1378.type=Plus	
\$1378.next=\$1380	
\$1380.type=ExpLink	
\$1380.next=\$1382	
\$1380.entry=\$1324	
\$1382.type=ExpLink	
\$1382.next=\$1384	
\$1382.entry=\$1352	
\$1384.type=ExpLink	
\$1384.entry=\$1356	
\$1434.type=SignedExpLink	
\$1434.entry=\$1428	
\$1434.sign=Minus	
\$1428.type=Min	
\$1428.formula=\$1418	
\$1418.type=List	
\$1418.next=\$1420	
\$1420.type=ObjLink	
\$1420.next=\$1422	
\$1420.entry=\$1324	
\$1422.type=ObjLink	
\$1422.next=\$1424	
\$1422.entry=\$1352	
\$1424.type=ObjLink	
\$1424.entry=\$1356	
VALUE(\$1342) = x	
VALUE(\$1354) = y	
VALUE(\$1358) = z	



Example 16.

$$\max_{k=1,\dots,n} x^{(k)}$$

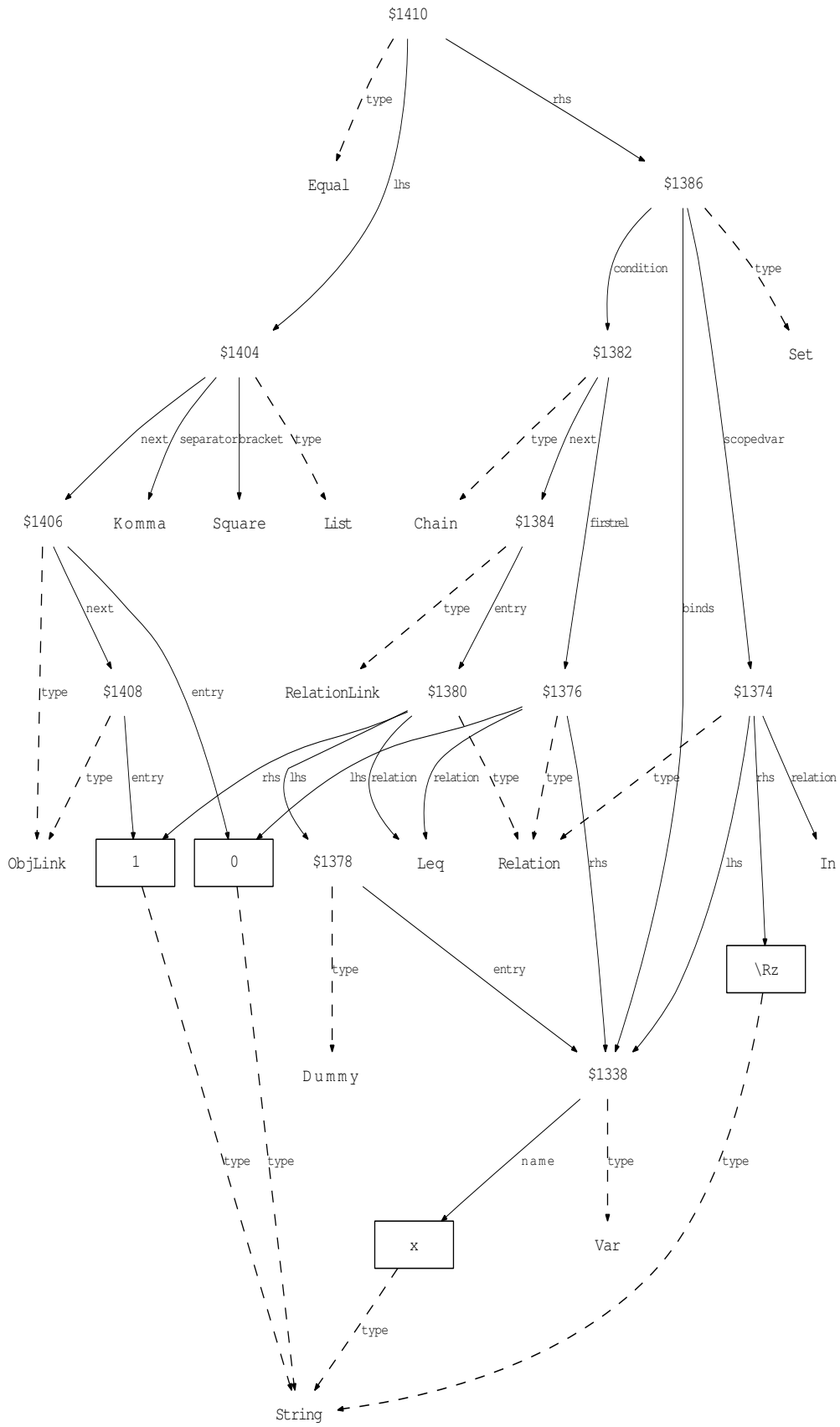
record	MATLAB construction code
\$1392.type=Max	x=createvar('x');
\$1392.formula=\$1364	k=createvar('k');
\$1392.binds=\$1352	n=createstring('n');
\$1392.index=\$1390	one=createstring('1');
\$1352.type=Var	dots=createstring('\ldots');
\$1352.name=\$1354	br=create('Bracket',k);
\$1354.type=String	in=create('Index',x,[],br);
\$1364.type=Index	komma=createname('Komma');
\$1364.formula=\$1324	none=createname('None');
\$1364.sup=\$1362	lst=create('List',{one,dots,n},komma,none);
\$1324.type=Var	eq=create('Equal',k,lst);
\$1324.name=\$1342	ex=create('Max',in,k,eq);
\$1342.type=String	
\$1362.type=Bracket	
\$1362.entry=\$1352	
\$1390.type=Equal	
\$1390.lhs=\$1352	
\$1390.rhs=\$1382	
\$1382.type=List	
\$1382.next=\$1384	
\$1382.separator=Komma	
\$1382.bracket=None	
\$1384.type=ObjLink	
\$1384.next=\$1386	
\$1384.entry=\$1358	
\$1358.type=String	
\$1386.type=ObjLink	
\$1386.next=\$1388	
\$1386.entry=\$1360	
\$1360.type=String	
\$1388.type=ObjLink	
\$1388.entry=\$1356	
\$1356.type=String	
VALUE(\$1342) = x	
VALUE(\$1354) = k	
VALUE(\$1356) = n	
VALUE(\$1358) = 1	
VALUE(\$1360) = \ldots	



**Example 17.** The interval on the LHS is a list with layout options for the parentheses.

$$[0, 1] = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$$

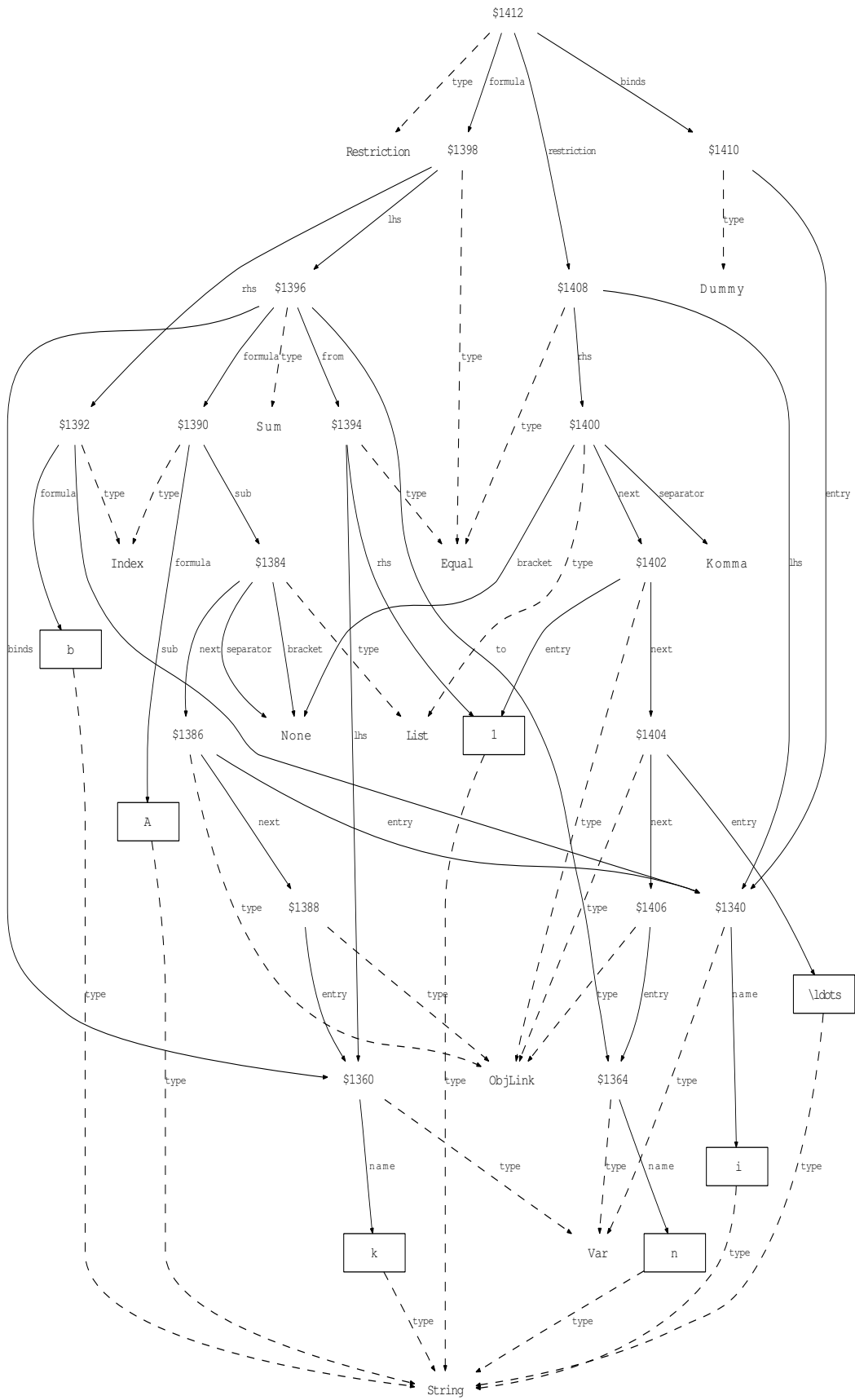
record	MATLAB construction code
\$1410.type=Equal	zero=createstring('0');
\$1410.lhs=\$1404	one=createstring('1');
\$1410.rhs=\$1386	Rz=createstring('\Rz');
\$1386.type=Set	x=createvar('x');
\$1386.condition=\$1382	in = createname('In');
\$1386.binds=\$1338	leq = createname('Leq');
\$1386.scopedvar=\$1374	xinRz=create('Relation',x,in,Rz);
\$1338.type=Var	leq1=create('Relation',zero,leq,x);
\$1338.name=\$1348	dummy=create('Dummy',x);
\$1348.type=String	leq2=create('Relation',dummy,leq,one);
\$1374.type=Relation	leq=create('Chain',leq1,{leq2});
\$1374.lhs=\$1338	rhs=create('Set',x,xinRz,leq);
\$1374.rhs=\$1336	komma=createname('Komma');
\$1374.relation=In	square=createname('Square');
\$1336.type=String	lhs=create('List',{zero,one},komma,square);
\$1382.type=Chain	ex=create('Equal',lhs,rhs);
\$1382.next=\$1384	
\$1382.firstrel=\$1376	
\$1376.type=Relation	
\$1376.lhs=\$1332	
\$1376.rhs=\$1338	
\$1376.relation=Leq	
\$1332.type=String	
\$1384.type=RelationLink	
\$1384.entry=\$1380	
\$1380.type=Relation	
\$1380.lhs=\$1378	
\$1380.rhs=\$1334	
\$1380.relation=Leq	
\$1334.type=String	
\$1378.type=Dummy	
\$1378.entry=\$1338	
\$1404.type=List	
\$1404.next=\$1406	
\$1404.separator=Komma	
\$1404.bracket=Square	
\$1406.type=ObjLink	
\$1406.next=\$1408	
\$1406.entry=\$1332	
\$1408.type=ObjLink	
\$1408.entry=\$1334	
VALUE(\$1332) = 0	
VALUE(\$1334) = 1	
VALUE(\$1336) = \Rz	
VALUE(\$1348) = x	



Example 18.

$$\sum_{k=1}^n A_{ik} = b_i \quad (i=1, \dots, n)$$

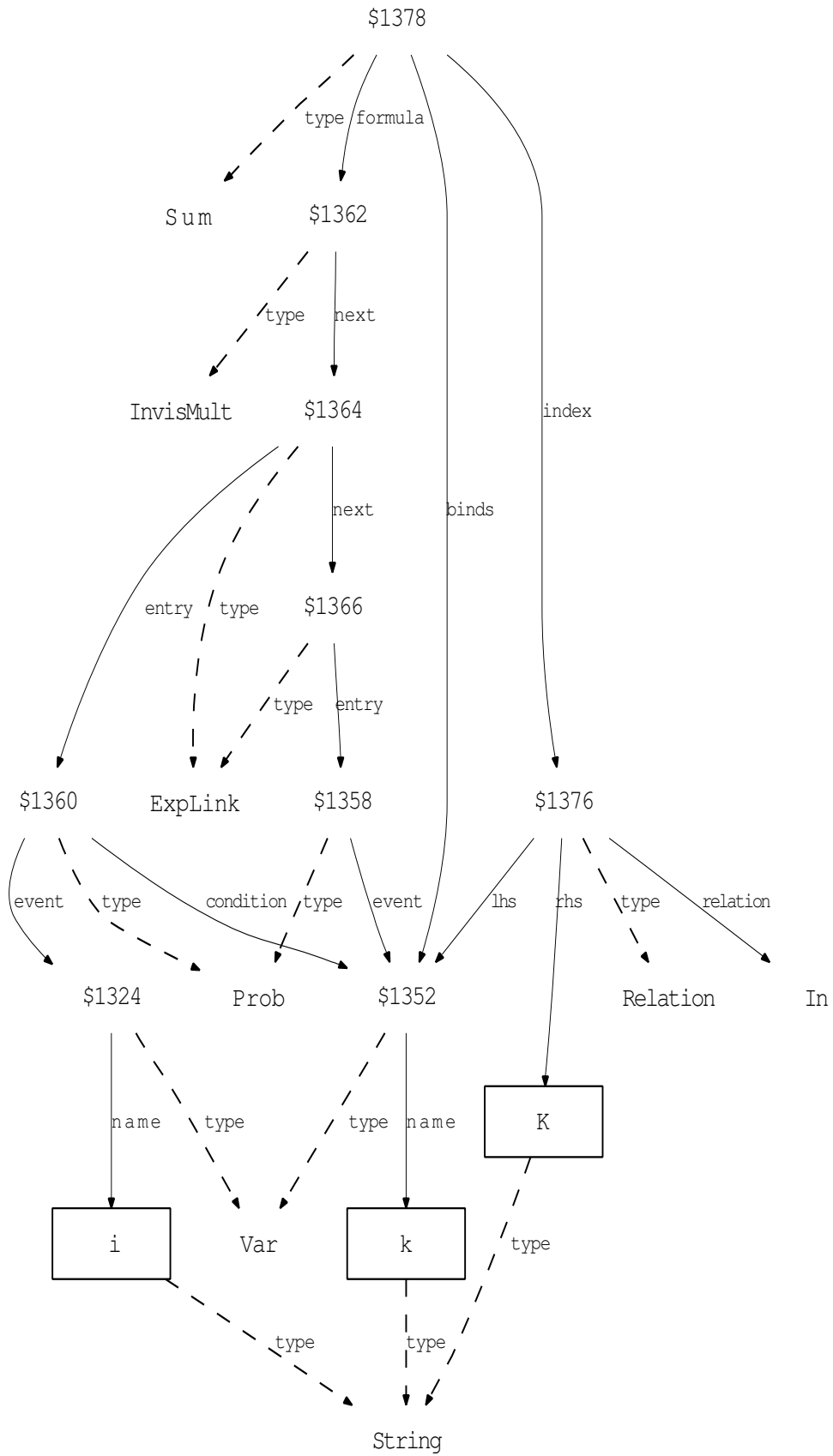
record	MATLAB construction code
\$1412.type=Restriction	A=createstring('A');
\$1412.formula=\$1398	b=createstring('b');
\$1412.binds=\$1410	one=createstring('1');
\$1412.restriction=\$1408	dots=createstring('\ldots');
\$1398.type=Equal	i=createvar('i');
\$1398.lhs=\$1396	k=createvar('k');
\$1398.rhs=\$1392	n=createvar('n');
\$1392.type=Index	none=createname('None');
\$1392.formula=\$1334	komma=createname('Komma');
\$1392.sub=\$1340	ik=create('List',{i,k},none,none);
\$1334.type=String	Aind=create('Index',A,ik);
\$1340.type=Var	bind=create('Index',b,i);
\$1340.name=\$1350	fromone=create('Equal',k,one);
\$1350.type=String	sum=create('Sum',Aind,k,[],fromone,n);
\$1396.type=Sum	eq=create('Equal',sum,bind);
\$1396.formula=\$1390	list=create('List',{one,dots,n},komma,none);
\$1396.binds=\$1360	restr=create('Equal',i,list);
\$1396.from=\$1394	dummy=create('Dummy',i);
\$1396.to=\$1364	ex=create('Restriction',eq,dummy,restr);
\$1360.type=Var	
\$1360.name=\$1362	
\$1362.type=String	
\$1364.type=Var	
\$1364.name=\$1366	
\$1366.type=String	
\$1390.type=Index	
\$1390.formula=\$1332	
\$1390.sub=\$1384	
\$1332.type=String	
\$1384.type=List	
\$1384.next=\$1386	
\$1384.separator=None	
\$1384.bracket=None	
\$1386.type=ObjLink	
\$1386.next=\$1388	
\$1386.entry=\$1340	
\$1388.type=ObjLink	
\$1388.entry=\$1360	
\$1394.type=Equal	
\$1394.lhs=\$1360	
\$1394.rhs=\$1336	
\$1336.type=String	
\$1408.type=Equal	
\$1408.lhs=\$1340	
\$1408.rhs=\$1400	
\$1400.type=List	
\$1400.next=\$1402	
\$1400.separator=Komma	
\$1400.bracket=None	
\$1402.type=ObjLink	
\$1402.next=\$1404	
\$1402.entry=\$1336	
\$1404.type=ObjLink	
\$1404.next=\$1406	
\$1404.entry=\$1338	
\$1338.type=String	
\$1406.type=ObjLink	
\$1406.entry=\$1364	
\$1410.type=Dummy	
\$1410.entry=\$1340	
VALUE(\$1332) = A	
VALUE(\$1334) = b	
VALUE(\$1336) = 1	
VALUE(\$1338) = \ldots	
VALUE(\$1350) = i	
VALUE(\$1362) = k	
VALUE(\$1366) = n	



**Example 19.**

$$\sum_{k \in K} \Pr(i|k) \Pr(k)$$

record	MATLAB construction code
\$1378.type=Sum	i=createvar('i');
\$1378.formula=\$1362	k=createvar('k');
\$1378.binds=\$1352	K=createstring('K');
\$1378.index=\$1376	prk=create('Prob',k);
\$1352.type=Var	prik=create('Prob',i,k);
\$1352.name=\$1354	mult=create('InvisMult',{prik,prk});
\$1354.type=String	inrel=createname('In');
\$1362.type=InvisMult	in=create('Relation',k,inrel,K);
\$1362.next=\$1364	ex=create('Sum',mult,k,in);
\$1364.type=ExpLink	
\$1364.next=\$1366	
\$1364.entry=\$1360	
\$1360.type=Prob	
\$1360.condition=\$1352	
\$1360.event=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1366.type=ExpLink	
\$1366.entry=\$1358	
\$1358.type=Prob	
\$1358.event=\$1352	
\$1376.type=Relation	
\$1376.lhs=\$1352	
\$1376.rhs=\$1356	
\$1376.relation=In	
\$1356.type=String	
VALUE(\$1342) = i	
VALUE(\$1354) = k	
VALUE(\$1356) = K	



## Example 20.

$$\|A\|_F := \sqrt{\sum_{\substack{i=1:m \\ k=1:n}} A_{ik}^2}$$

record	MATLAB construction code
\$1432.type=EqualByDef	i=createvar('i');
\$1432.lhs=\$1428	k=createvar('k');
\$1432.rhs=\$1430	m=createvar('m');
\$1428.type=Norm	n=createvar('n');
\$1428.formula=\$1364	A=createstring('A');
\$1428.index=\$1366	F=createstring('F');
\$1364.type=String	one=createstring('1');
\$1366.type=String	two=createstring('2');
\$1430.type=Sqrt	none=createname('none');
\$1430.radicand=\$1426	colon=createname('colon');
\$1426.type=Sum	ik=create('List',{i,k},none,none);
\$1426.formula=\$1396	Aik=create('Index',A,ik);
\$1426.binds=\$1420	Aik2=create('Power',Aik,two);
\$1426.index=\$1414	onem=create('List',{one,m},colon,none);
\$1396.type=Power	onen=create('List',{one,n},colon,none);
\$1396.base=\$1394	irestr=create('Equal',i,onem);
\$1396.exponent=\$1370	krestr=create('Equal',k,onen);
\$1370.type=String	indices=create('List',{irestr,krestr});
\$1394.type=Index	ik=create('VarList',{i,k});
\$1394.formula=\$1364	sum=create('Sum',Aik2,ik,indices);
\$1394.sub=\$1388	norm=create('Norm',A,F);
\$1388.type=List	sqr=create('Sqrt',sum);
\$1388.next=\$1390	ex=create('EqualByDef',norm,sqr);
\$1388.separator=none	
\$1388.bracket=none	
\$1390.type=ObjLink	
\$1390.next=\$1392	
\$1390.entry=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1392.type=ObjLink	
\$1392.next=\$1352	
\$1392.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1414.type=List	
\$1414.next=\$1416	
\$1416.type=ObjLink	
\$1416.next=\$1418	
\$1416.entry=\$1410	
\$1410.type=Equal	
\$1410.lhs=\$1324	
\$1410.rhs=\$1398	
\$1398.type=List	
\$1398.next=\$1400	
\$1398.separator=colon	
\$1398.bracket=none	
\$1400.type=ObjLink	
\$1400.next=\$1402	
\$1400.entry=\$1368	
\$1368.type=String	
\$1402.type=ObjLink	
\$1402.entry=\$1356	
\$1356.type=Var	
\$1356.name=\$1358	
\$1358.type=String	
\$1418.type=ObjLink	
\$1418.entry=\$1412	
\$1412.type=Equal	
\$1412.lhs=\$1352	
\$1412.rhs=\$1404	
\$1404.type=List	
\$1404.next=\$1406	
\$1404.separator=colon	
\$1404.bracket=none	
\$1406.type=ObjLink	
\$1406.next=\$1408	
\$1406.entry=\$1368	
\$1408.type=ObjLink	
\$1408.entry=\$1360	
\$1360.type=Var	
\$1360.name=\$1362	
\$1362.type=String	
\$1420.type=VarList	
\$1420.next=\$1422	
\$1422.type=VarLink	
\$1422.next=\$1424	
\$1422.entry=\$1324	
\$1424.type=VarLink	
\$1424.entry=\$1352	
VALUE(\$1342) = i	
VALUE(\$1354) = k	
VALUE(\$1358) = m	
VALUE(\$1362) = n	
VALUE(\$1364) = A	
VALUE(\$1366) = F	
VALUE(\$1368) = 1	
VALUE(\$1370) = 2	



**Example 21.**

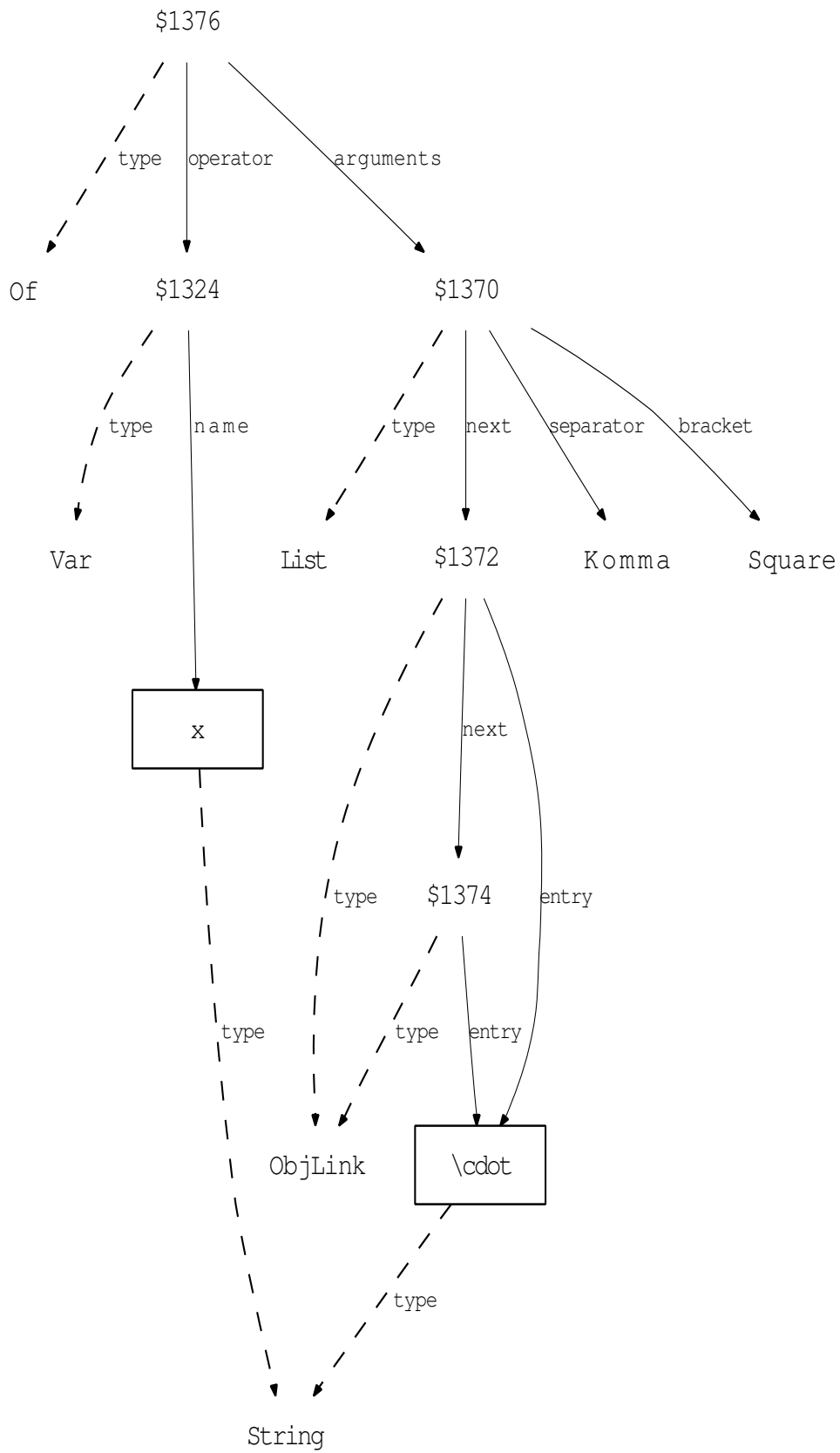
$$x^l \quad (l=1:n)$$

record	MATLAB construction code
\$1404.type=Restriction	x=createvar('x');
\$1404.formula=\$1396	n=createvar('n');
\$1404.binds=\$1356	l=createvar('l');
\$1404.restriction=\$1402	one=createint('1');
\$1356.type=Var	none=createname('none');
\$1356.name=\$1358	colon=createname('colon');
\$1358.type=String	onen=create('List',{one,n},colon,none);
\$1396.type=Alternative	pow=create('Power',x,l);
\$1396.next=\$1398	ind=create('Index',x,[],l);
\$1398.type=ObjectLink	xl=create('Alternative',{pow,ind});
\$1398.next=\$1400	equ=create('Equal',l,onen);
\$1398.entry=\$1392	ex=create('Restriction',xl,l,equ);
\$1392.type=Power	
\$1392.base=\$1324	
\$1392.exponent=\$1356	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1400.type=ObjectLink	
\$1400.entry=\$1394	
\$1394.type=Index	
\$1394.formula=\$1324	
\$1394.sup=\$1356	
\$1402.type=Equal	
\$1402.lhs=\$1356	
\$1402.rhs=\$1386	
\$1386.type=List	
\$1386.next=\$1388	
\$1386.separator=colon	
\$1386.bracket=none	
\$1388.type=ObjLink	
\$1388.next=\$1390	
\$1388.entry=\$1368	
\$1368.type=Integer	
\$1390.type=ObjLink	
\$1390.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
VALUE(\$1342) = x	
VALUE(\$1354) = n	
VALUE(\$1358) = 1	
VALUE(\$1368) = 1	



**Example 22.** $x[\cdot, \cdot]$ 

record	MATLAB construction code
\$1376.type=Of	<code>x=createvar('x');</code>
\$1376.operator=\$1324	<code>dot=createstring('\cdot');</code>
\$1376.arguments=\$1370	<code>komma = createname('Komma');</code>
\$1324.type=Var	<code>square = createname('Square');</code>
\$1324.name=\$1342	<code>args=create('List',{dot,dot},komma,square);</code>
\$1342.type=String	<code>ex=create('Of',x,args);</code>
\$1370.type=List	
\$1370.next=\$1372	
\$1370.separator=Komma	
\$1370.bracket=Square	
\$1372.type=ObjLink	
\$1372.next=\$1374	
\$1372.entry=\$1352	
\$1352.type=String	
\$1374.type=ObjLink	
\$1374.entry=\$1352	
VALUE(\$1342) = x	
VALUE(\$1352) = \cdot	



**Example 23.**

$$K_{B_{ij}}^2 = {}^2\Phi^{i,j}$$

record	MATLAB construction code
\$1404.type=Equal	i=createvar('i');
\$1404.lhs=\$1402	j=createvar('j');
\$1404.rhs=\$1396	K=createstring('K');
\$1396.type=Index	B=createstring('B');
\$1396.formula=\$1360	phi=createstring('\Phi');
\$1396.sup=\$1388	two=createint('2');
\$1396.lsup=\$1370	komma=createstring(',');
\$1360.type=String	none=createnam('None');
\$1370.type=Integer	ij=create('List',{i,j},none,none);
\$1388.type=List	ikommaj=create('List',{i,komma,j},none,none);
\$1388.next=\$1390	phiindex=create('Index',phi,[],ikommaj,two);
\$1388.separator=None	Bindex=create('Index',B,ij);
\$1388.bracket=None	Kindex=create('Index',K,Bindex);
\$1390.type=ObjLink	Ksq=create('Power',Kindex,two);
\$1390.next=\$1392	ex=create('Equal',Ksq,phiindex);
\$1390.entry=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1392.type=ObjLink	
\$1392.next=\$1394	
\$1392.entry=\$1372	
\$1372.type=String	
\$1394.type=ObjLink	
\$1394.entry=\$1352	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1402.type=Power	
\$1402.base=\$1400	
\$1402.exponent=\$1370	
\$1400.type=Index	
\$1400.formula=\$1356	
\$1400.sub=\$1398	
\$1356.type=String	
\$1398.type=Index	
\$1398.formula=\$1358	
\$1398.sub=\$1382	
\$1358.type=String	
\$1382.type=List	
\$1382.next=\$1384	
\$1382.separator=None	
\$1382.bracket=None	
\$1384.type=ObjLink	
\$1384.next=\$1386	
\$1384.entry=\$1324	
\$1386.type=ObjLink	
\$1386.entry=\$1352	
VALUE(\$1342) = i	
VALUE(\$1354) = j	
VALUE(\$1356) = K	
VALUE(\$1358) = B	
VALUE(\$1360) = \Phi	
VALUE(\$1370) = 2	
VALUE(\$1372) = ,	



**Example 24.**

$$\int_B \int_A f(x, y) dx dy = \int_{A \times B} f(x, y) dxy$$

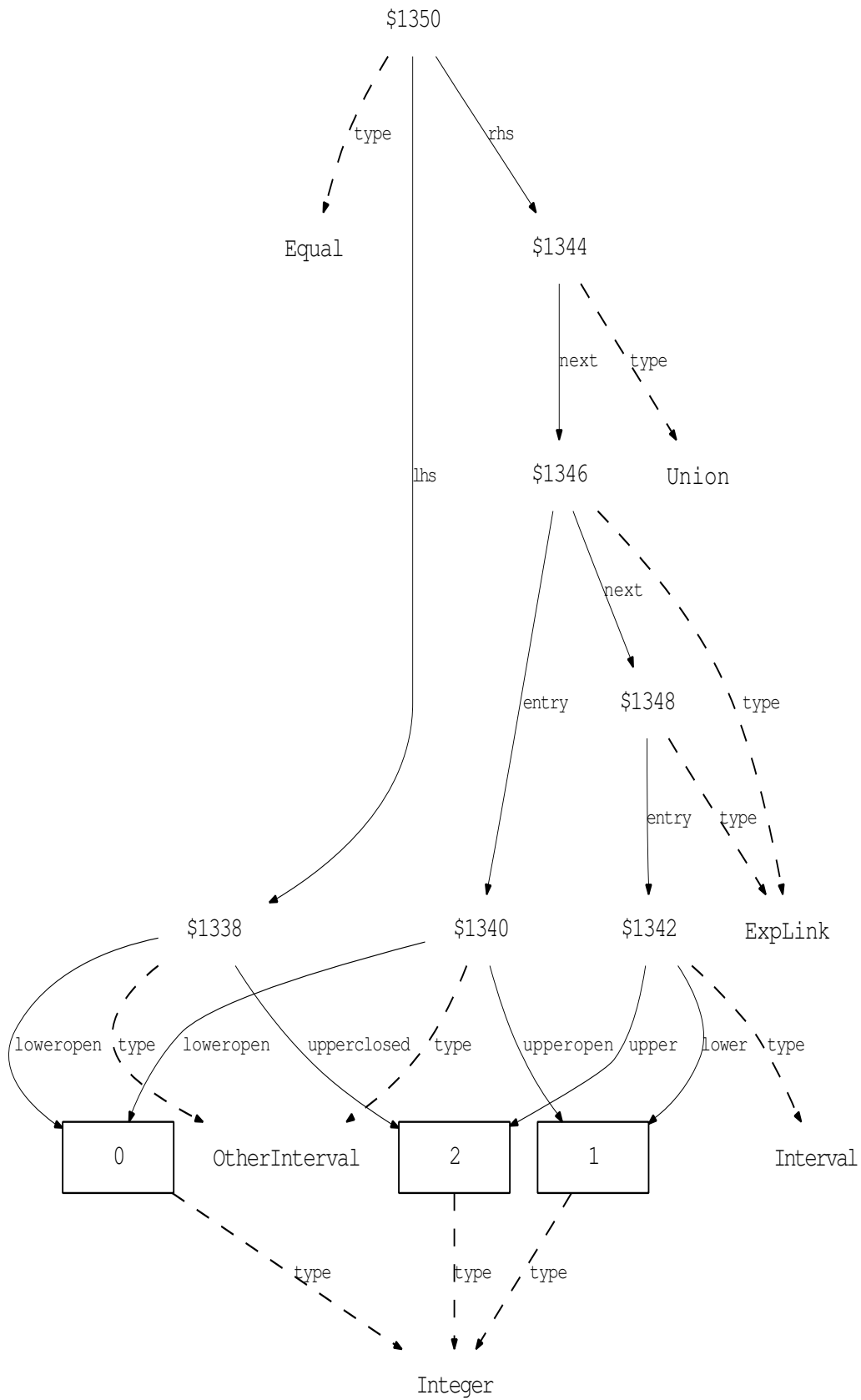
record	MATLAB construction code
\$1396.type=Equal	x=createvar('x');
\$1396.lhs=\$1372	y=createvar('y');
\$1396.rhs=\$1394	f=createstring('f');
\$1372.type=Int	A=createstring('A');
\$1372.formula=\$1370	B=createstring('B');
\$1372.binds=\$1352	args=create('List',{x,y});
\$1372.index=\$1360	fx=create('Of',f,args);
\$1372.variable=\$1352	lhsinner=create('Int',fx,x,x,A);
\$1352.type=Var	lhs=create('Int',lhsinner,y,y,B);
\$1352.name=\$1354	AtimesB=create('Times',{A,B});
\$1354.type=String	none=createnam('None');
\$1360.type=String	xy=create('List',{x,y},none,none);
\$1370.type=Int	rhs=create('Int',fx,xy,xy,AtimesB);
\$1370.formula=\$1368	ex=create('Equal',lhs,rhs);
\$1370.binds=\$1324	
\$1370.index=\$1358	
\$1370.variable=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1358.type=String	
\$1368.type=Of	
\$1368.operator=\$1356	
\$1368.arguments=\$1362	
\$1356.type=String	
\$1362.type=List	
\$1362.next=\$1364	
\$1364.type=ObjLink	
\$1364.next=\$1366	
\$1364.entry=\$1324	
\$1366.type=ObjLink	
\$1366.entry=\$1352	
\$1394.type=Int	
\$1394.formula=\$1368	
\$1394.binds=\$1388	
\$1394.index=\$1374	
\$1394.variable=\$1388	
\$1374.type=Times	
\$1374.next=\$1376	
\$1376.type=ExpLink	
\$1376.next=\$1378	
\$1376.entry=\$1358	
\$1378.type=ExpLink	
\$1378.entry=\$1360	
\$1388.type=List	
\$1388.next=\$1390	
\$1388.separator=None	
\$1388.bracket=None	
\$1390.type=ObjLink	
\$1390.next=\$1392	
\$1390.entry=\$1324	
\$1392.type=ObjLink	
\$1392.entry=\$1352	
VALUE(\$1342) = x	
VALUE(\$1354) = y	
VALUE(\$1356) = f	
VALUE(\$1358) = A	
VALUE(\$1360) = B	



**Example 25.**

$$(0, 2] = (0, 1) \cup [1, 2]$$

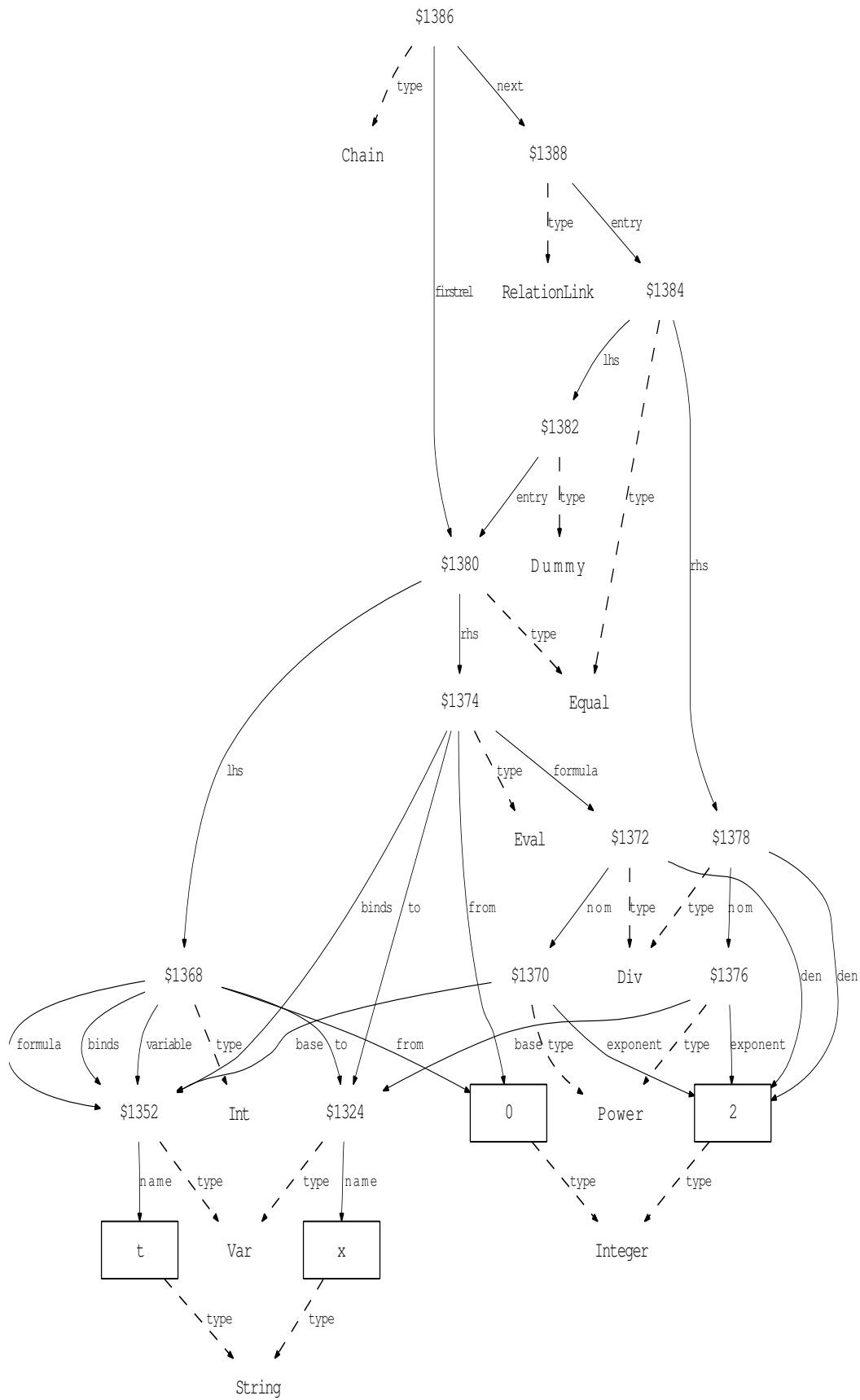
record	MATLAB construction code
\$1350.type=Equal	zero=createint('0');
\$1350.lhs=\$1338	one=createint('1');
\$1350.rhs=\$1344	two=createint('2');
\$1338.type=OtherInterval	lhs=create('OtherInterval', [], zero, [], two);
\$1338.loweropen=\$1332	rhs1=create('OtherInterval', [], zero, one);
\$1338.upperclosed=\$1336	rhs2=create('Interval', one, two);
\$1332.type=Integer	rhs=create('Union', {rhs1, rhs2});
\$1336.type=Integer	ex=create('Equal', lhs, rhs);
\$1344.type=Union	
\$1344.next=\$1346	
\$1346.type=Explink	
\$1346.next=\$1348	
\$1346.entry=\$1340	
\$1340.type=OtherInterval	
\$1340.loweropen=\$1332	
\$1340.upperopen=\$1334	
\$1334.type=Integer	
\$1348.type=Explink	
\$1348.entry=\$1342	
\$1342.type=Interval	
\$1342.lower=\$1334	
\$1342.upper=\$1336	
VALUE(\$1332) = 0	
VALUE(\$1334) = 1	
VALUE(\$1336) = 2	



**Example 26.**

$$\int_0^x t \, dt = \frac{t^2}{2} \Big|_0^x = \frac{x^2}{2}$$

record	MATLAB construction code
\$1386.type=Chain	x=createvar('x');
\$1386.next=\$1388	t=createvar('t');
\$1386.firstrel=\$1380	zero=createint('0');
\$1380.type=Equal	two=createint('2');
\$1380.lhs=\$1368	int=create('Int',t,t,t,[],zero,x);
\$1380.rhs=\$1374	tsq=create('Power',t,two);
\$1368.type=Int	tsqh=create('Div',tsq,two);
\$1368.formula=\$1352	teval=create('Eval',tsqh,t,[],zero,x);
\$1368.binds=\$1352	xsq=create('Power',x,two);
\$1368.from=\$1364	xsqh=create('Div',xsq,two);
\$1368.to=\$1324	eqleft=create('Equal',int,teval);
\$1368.variable=\$1352	dummy=create('Dummy',eqleft);
\$1324.type=Var	eqrigh=create('Equal',dummy,xsqh);
\$1324.name=\$1342	ex=create('Chain',eqleft,{eqrigh});
\$1342.type=String	
\$1352.type=Var	
\$1352.name=\$1354	
\$1354.type=String	
\$1364.type=Integer	
\$1374.type=Eval	
\$1374.formula=\$1372	
\$1374.binds=\$1352	
\$1374.from=\$1364	
\$1374.to=\$1324	
\$1372.type=Div	
\$1372.nom=\$1370	
\$1372.den=\$1366	
\$1366.type=Integer	
\$1370.type=Power	
\$1370.base=\$1352	
\$1370.exponent=\$1366	
\$1388.type=RelationLink	
\$1388.entry=\$1384	
\$1384.type=Equal	
\$1384.lhs=\$1382	
\$1384.rhs=\$1378	
\$1378.type=Div	
\$1378.nom=\$1376	
\$1378.den=\$1366	
\$1376.type=Power	
\$1376.base=\$1324	
\$1376.exponent=\$1366	
\$1382.type=Dummy	
\$1382.entry=\$1380	
VALUE(\$1342) = x	
VALUE(\$1354) = t	
VALUE(\$1364) = 0	
VALUE(\$1366) = 2	



Example 27.

$$A_{;k}^i = A_{,k}^i + A_{,k}^i \Gamma_{ka}^i$$

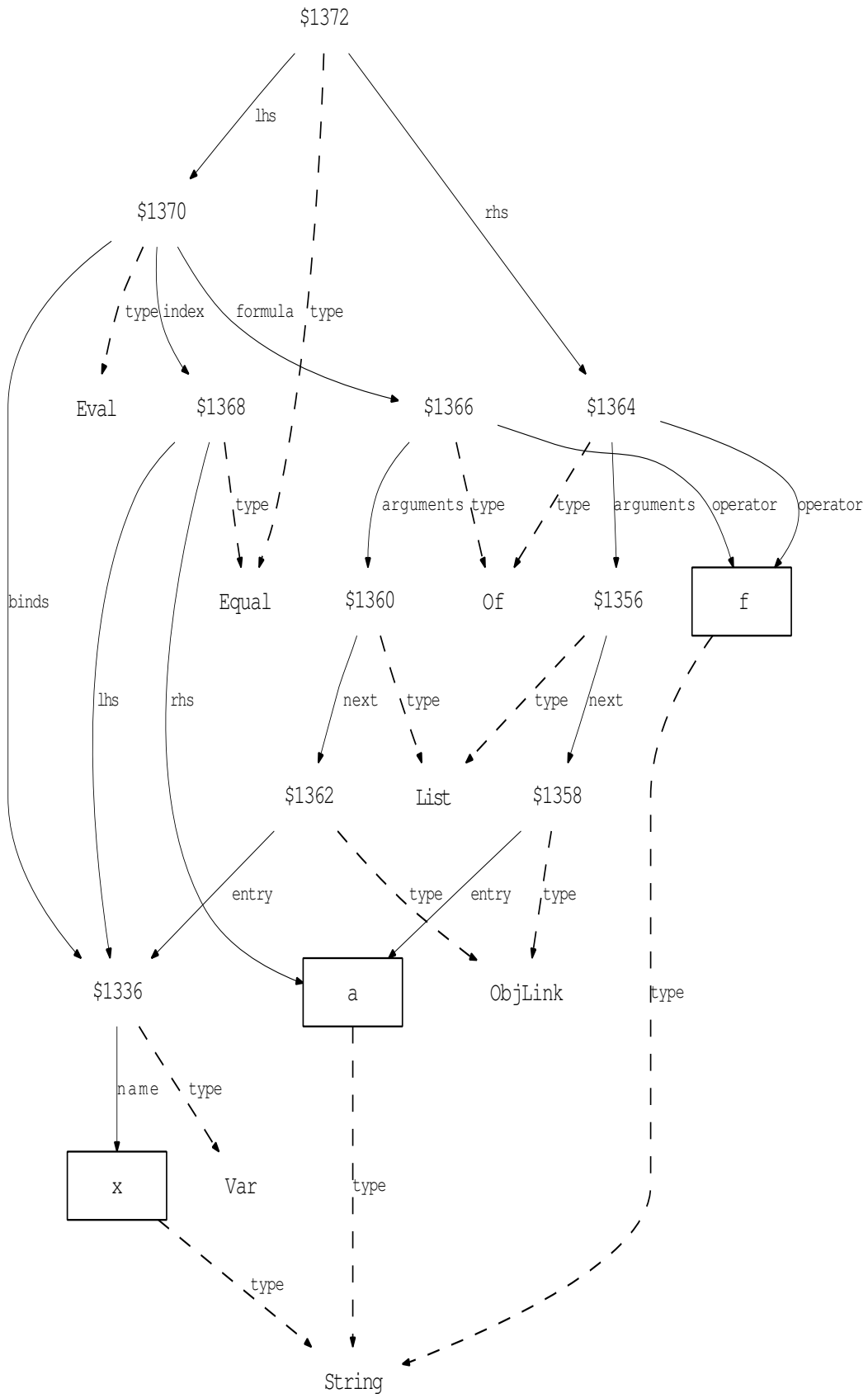
record	MATLAB construction code
\$1428.type=Equal	A=createstring('A');
\$1428.lhs=\$1408	G=createstring('\Gamma');
\$1428.rhs=\$1422	semik=createstring(';');
\$1408.type=Index	komma=createstring(',');
\$1408.formula=\$1332	blank=createstring(' ');
\$1408.sub=\$1378	i=createvar('i');
\$1408.sup=\$1342	k=createvar('k');
\$1332.type=String	a=createvar('a');
\$1342.type=Var	none=createname('None');
\$1342.name=\$1352	bsk=create('List', {blank, semik, k}, none, none);
\$1352.type=String	bkk=create('List', {blank, komma, k}, none, none);
\$1378.type=List	bka=create('List', {blank, k, a}, none, none);
\$1378.next=\$1380	ka=create('List', {k, a}, none, none);
\$1378.separator=None	lhs=create('Index', A, bsk, i);
\$1378.bracket=None	rhs1=create('Index', A, bkk, i);
\$1380.type=ObjLink	rhs2=create('Index', G, ka, i);
\$1380.next=\$1382	rhs3=create('Index', A, [], a);
\$1380.entry=\$1340	mult=create('InvisMult', {rhs1, rhs2});
\$1340.type=String	rhs=create('Plus', {rhs1, mult});
\$1382.type=ObjLink	ex=create('Equal', lhs, rhs);
\$1382.next=\$1384	
\$1382.entry=\$1336	
\$1336.type=String	
\$1384.type=ObjLink	
\$1384.entry=\$1362	
\$1362.type=Var	
\$1362.name=\$1364	
\$1364.type=String	
\$1422.type=Plus	
\$1422.next=\$1424	
\$1424.type=ExpLink	
\$1424.next=\$1426	
\$1424.entry=\$1410	
\$1410.type=Index	
\$1410.formula=\$1332	
\$1410.sub=\$1386	
\$1410.sup=\$1342	
\$1386.type=List	
\$1386.next=\$1388	
\$1386.separator=None	
\$1386.bracket=None	
\$1388.type=ObjLink	
\$1388.next=\$1390	
\$1388.entry=\$1340	
\$1390.type=ObjLink	
\$1390.next=\$1392	
\$1390.entry=\$1338	
\$1338.type=String	
\$1392.type=ObjLink	
\$1392.entry=\$1362	
\$1426.type=ExpLink	
\$1426.entry=\$1416	
\$1416.type=InvisMult	
\$1416.next=\$1418	
\$1418.type=ExpLink	
\$1418.next=\$1420	
\$1418.entry=\$1410	
\$1420.type=ExpLink	
\$1420.entry=\$1412	
\$1412.type=Index	
\$1412.formula=\$1334	
\$1412.sub=\$1402	
\$1412.sup=\$1342	
\$1334.type=String	
\$1402.type=List	
\$1402.next=\$1404	
\$1402.separator=None	
\$1402.bracket=None	
\$1404.type=ObjLink	
\$1404.next=\$1406	
\$1404.entry=\$1362	
\$1406.type=ObjLink	
\$1406.entry=\$1366	
\$1366.type=Var	
\$1366.name=\$1368	
\$1368.type=String	
VALUE(\$1332) = A	
VALUE(\$1334) = \Gamma	
VALUE(\$1336) = ;	
VALUE(\$1338) = ,	
VALUE(\$1340) =	
VALUE(\$1352) = i	
VALUE(\$1364) = k	
VALUE(\$1368) = a	



**Example 28.**

$$f(x)|_{x=a} = f(a)$$

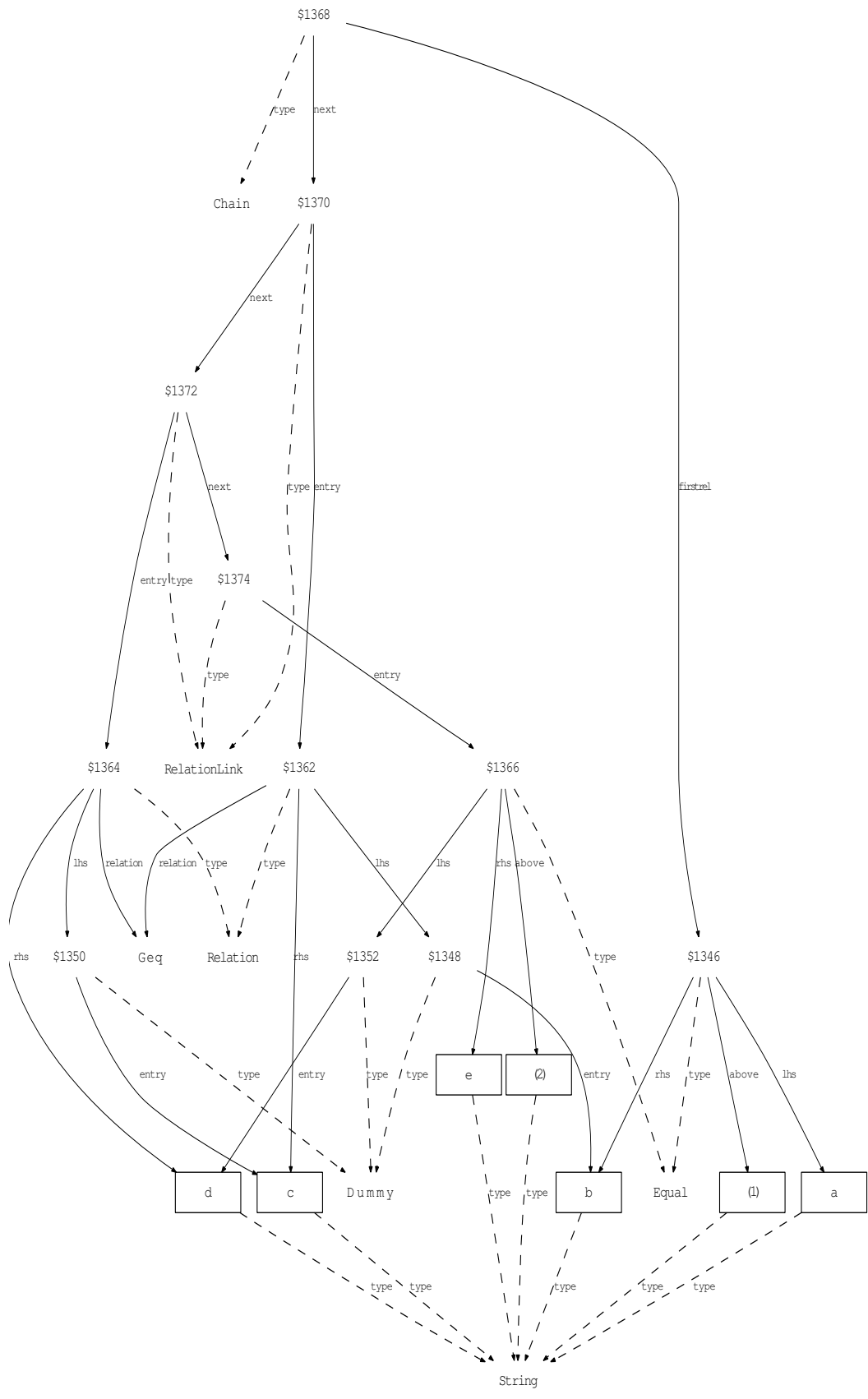
record	MATLAB construction code
\$1372.type=Equal	a=createstring('a');
\$1372.lhs=\$1370	f=createstring('f');
\$1372.rhs=\$1364	x=createvar('x');
\$1364.type=Of	arga=create('List',{a});
\$1364.operator=\$1334	argx=create('List',{x});
\$1364.arguments=\$1356	fa=create('Of',f,arga);
\$1334.type=String	fx=create('Of',f,argx);
\$1356.type=List	xeqa=create('Equal',x,a);
\$1356.next=\$1358	feval=create('Eval',fx,x,xeqa);
\$1358.type=ObjLink	ex=create('Equal',feval,fa);
\$1358.entry=\$1332	
\$1332.type=String	
\$1370.type=Eval	
\$1370.formula=\$1366	
\$1370.binds=\$1336	
\$1370.index=\$1368	
\$1336.type=Var	
\$1336.name=\$1346	
\$1346.type=String	
\$1366.type=Of	
\$1366.operator=\$1334	
\$1366.arguments=\$1360	
\$1360.type=List	
\$1360.next=\$1362	
\$1362.type=ObjLink	
\$1362.entry=\$1336	
\$1368.type=Equal	
\$1368.lhs=\$1336	
\$1368.rhs=\$1332	
VALUE(\$1332) = a	
VALUE(\$1334) = f	
VALUE(\$1346) = x	



**Example 29.**

$$a \stackrel{(1)}{=} b \geq c \geq d \stackrel{(2)}{=} e$$

record	MATLAB construction code
\$1368.type=Chain	a=createstring('a');
\$1368.next=\$1370	b=createstring('b');
\$1368.firstrel=\$1346	c=createstring('c');
\$1346.type=Equal	d=createstring('d');
\$1346.lhs=\$1332	e=createstring('e');
\$1346.rhs=\$1334	abone=createstring('(1)');
\$1346.above=\$1342	abtwo=createstring('(2)');
\$1332.type=String	ab=create('Equal',a,b,abone);
\$1334.type=String	dummyb=create('Dummy',b);
\$1342.type=String	dummyc=create('Dummy',c);
\$1370.type=RelationLink	dummyd=create('Dummy',d);
\$1370.next=\$1372	geq=createnamename('Geq');
\$1370.entry=\$1362	linkc=create('Relation',dummyb,geq,c);
\$1362.type=Relation	linkd=create('Relation',dummyc,geq,d);
\$1362.lhs=\$1348	linke=create('Equal',dummyd,e,abtwo);
\$1362.rhs=\$1336	ex=create('Chain',ab,{linkc,linkd,linke});
\$1362.relation=Geq	
\$1336.type=String	
\$1348.type=Dummy	
\$1348.entry=\$1334	
\$1372.type=RelationLink	
\$1372.next=\$1374	
\$1372.entry=\$1364	
\$1364.type=Relation	
\$1364.lhs=\$1350	
\$1364.rhs=\$1338	
\$1364.relation=Geq	
\$1338.type=String	
\$1350.type=Dummy	
\$1350.entry=\$1336	
\$1374.type=RelationLink	
\$1374.entry=\$1366	
\$1366.type=Equal	
\$1366.lhs=\$1352	
\$1366.rhs=\$1340	
\$1366.above=\$1344	
\$1340.type=String	
\$1344.type=String	
\$1352.type=Dummy	
\$1352.entry=\$1338	
VALUE(\$1332) = a	
VALUE(\$1334) = b	
VALUE(\$1336) = c	
VALUE(\$1338) = d	
VALUE(\$1340) = e	
VALUE(\$1342) = (1)	
VALUE(\$1344) = (2)	



**Example 30.**

$$\{x \in X(k) \mid f(x, k)=0\}$$

record	MATLAB construction code
\$1396.type=Set	x=createvar('x');
\$1396.condition=\$1378	X=createstring('X');
\$1396.binds=\$1324	k=createvar('k');
\$1396.scopedvar=\$1394	f=createstring('f');
\$1324.type=Var	zero=createint('0');
\$1324.name=\$1342	xklist=create('List',{x,k});
\$1342.type=String	fkt=create('Of',f,xklist);
\$1378.type=Equal	rhs=create('Equal',fkt,zero);
\$1378.lhs=\$1376	klist=create('List',{k});
\$1378.rhs=\$1368	scopeset=create('Of',X,klist);
\$1368.type=Integer	in=createnam('In');
\$1376.type=Of	scope=create('Relation',x,in,scopeset);
\$1376.operator=\$1358	ex=create('Set',x,scope,rhs);
\$1376.arguments=\$1370	
\$1358.type=String	
\$1370.type=List	
\$1370.next=\$1372	
\$1372.type=ObjLink	
\$1372.next=\$1374	
\$1372.entry=\$1324	
\$1374.type=ObjLink	
\$1374.entry=\$1354	
\$1354.type=Var	
\$1354.name=\$1356	
\$1356.type=String	
\$1394.type=Relation	
\$1394.lhs=\$1324	
\$1394.rhs=\$1384	
\$1394.relation=In	
\$1384.type=Of	
\$1384.operator=\$1352	
\$1384.arguments=\$1380	
\$1352.type=String	
\$1380.type=List	
\$1380.next=\$1382	
\$1382.type=ObjLink	
\$1382.entry=\$1354	
VALUE(\$1342) = x	
VALUE(\$1352) = X	
VALUE(\$1356) = k	
VALUE(\$1358) = f	
VALUE(\$1368) = 0	



**Example 31.**

$$X(k) \mid \lambda x.f(x, k)=0$$

record	MATLAB construction code
\$1390.type=Mid	x=createvar('x');
\$1390.lhs=\$1384	X=createstring('X');
\$1390.rhs=\$1388	k=createvar('k');
\$1384.type=Of	f=createstring('f');
\$1384.operator=\$1352	zero=createint('0');
\$1384.arguments=\$1380	xklist=create('List',{x,k});
\$1352.type=String	fkt=create('Of',f,xklist);
\$1380.type=List	rhs=create('Equal',fkt,zero);
\$1380.next=\$1382	klist=create('List',{k});
\$1382.type=ObjLink	scopeset=create('Of',X,klist);
\$1382.entry=\$1354	dummy=create('Dummy',x);
\$1354.type=Var	lam=create('Lambda',rhs,x,x);
\$1354.name=\$1356	ex=create('Mid',scopeset,lam);
\$1356.type=String	
\$1388.type=Lambda	
\$1388.formula=\$1378	
\$1388.binds=\$1324	
\$1388.variable=\$1324	
\$1324.type=Var	
\$1324.name=\$1342	
\$1342.type=String	
\$1378.type=Equal	
\$1378.lhs=\$1376	
\$1378.rhs=\$1368	
\$1368.type=Integer	
\$1376.type=Of	
\$1376.operator=\$1358	
\$1376.arguments=\$1370	
\$1358.type=String	
\$1370.type=List	
\$1370.next=\$1372	
\$1372.type=ObjLink	
\$1372.next=\$1374	
\$1372.entry=\$1324	
\$1374.type=ObjLink	
\$1374.entry=\$1354	
VALUE(\$1342) = x	
VALUE(\$1352) = X	
VALUE(\$1356) = k	
VALUE(\$1358) = f	
VALUE(\$1368) = 0	



**Acknowledgements.** We thank the members of the FMathL seminar, in particular Hermann Schichl and Kevin Kofler for their contributions.

Support by the Austrian Science Fund (FWF) under contract number P20631 is gratefully acknowledged.

## References

- [1] P. Schodl and A. Neumaier. The FMathL type system. *Manuscript*, 2010.